Available online at www.sciencedirect.com
ScienceDirect
Theoretical Computer Science 373 (2007) 70–91
Theoretical Computer Science
www.elsevier.com/locate/tcs

Reflection in membership equational logic, many-sorted equational logic, Horn logic with equality, and rewriting logic☆

Manuel Clavel[a,*], José Meseguer[b], Miguel Palomino[a]

[a] *Departamento de Sistemas Informáticos y Computación, Universidad Complutense de Madrid, C/ Profesor García Santesmases, 28040, Madrid, Spain*

[b] *Computer Science Department, University of Illinois at Urbana-Champaign, Thomas M. Siebel Center for Computer Science, 201 N. Goodwin, Urbana, IL 61801-2302, United States*

Received 27 January 2005; received in revised form 11 December 2006; accepted 15 December 2006

Communicated by U. Montanari

Abstract

We show that the generalized variant of formal systems where the underlying equational specifications are membership equational theories, and where the rules are conditional and can have equations, memberships and rewrites in the conditions is reflective. We also show that membership equational logic, many-sorted equational logic, and Horn logic with equality are likewise reflective. These results provide logical foundations for reflective languages and tools based on these logics.

Keywords: Reflection; Reflective logics; Reflective programming languages; Universal theories; Rewriting logic; Membership equational logic; Maude

1. Introduction

Reflection is a very powerful and useful feature of rewriting logic. Intuitively, from a logical viewpoint reflection consists in the capacity of a system for reasoning about important aspects of its own metalanguage; from a computational viewpoint, it means that programs can become data that can be manipulated and executed by other programs. Classical examples where reflection (in this intuitive sense) can be seen in action are the coding of first-order arithmetic in itself by Gödel and the universal machine of Turing. More recently, many computer scientists have recognized the power and usefulness of reflection in areas such as programming languages [53,51,33,32,1], theorem proving [6,46,30], as well as metalevel architectures, distributed computation, program transformation, and databases [39,19,54,21,8].

☆ This work is an extended and revised version of a paper presented at WRLA 2002, including the detailed proofs of all the results. Research was supported by ONR Grant N00014-02-1-0715, NSF Grant CCR-0234524, and by DARPA through Air Force Research Laboratory Contract F30602-02-C-0130; and by the Spanish projects MELODIAS TIC 2002-01167 and MIDAS TIC 2003-0100.

* Corresponding author. Tel.: +34 913947555; fax: +34 913947529.
E-mail addresses: clavel@sip.ucm.es (M. Clavel), meseguer@cs.uiuc.edu (J. Meseguer), miguelpt@sip.ucm.es (M. Palomino).

0304-3975/$ - see front matter
doi:10.1016/j.tcs.2006.12.009

The study of reflection in logics with good computational properties is particularly interesting since both aspects of reflection—the logical and the computational one—are involved. Rewriting logic [41] is a logic of concurrent change that can naturally deal with state and with highly nondeterministic concurrent computations. Rewriting logic is parameterized with respect to the version of an underlying equational logic, which can be unsorted, many-sorted, order-sorted, or the recently developed membership equational logic [43]. The signature of a rewrite theory describes the structure for the states of a system, and the rewrite rules (that may be conditional) describe which elementary local transitions are possible in the distributed state. The rewriting logic research program has shown good signs of vitality, including six international workshops [42,34,27,29,35,20] and three programming language implementation efforts, namely ELAN [4] in France, CafeOBJ [28] in Japan, and Maude [11,13] in the USA.

With respect to reflection, Clavel and Meseguer have formerly given detailed proofs for increasingly general fragments of rewriting logic, namely: (1) unsorted and unconditional [10], (2) unsorted conditional [17]; and (3) many-sorted conditional [17]. This paper generalizes these previous results to the case of conditional rewrite theories whose underlying equational specifications are theories in *membership equational logic* [43]. Conditional rules in this latter case are very general, since they can involve not only other rewrites, but also equations and memberships as conjuncts. The work presented here is also related to Palomino's own research on rewriting logic reflection [45].

But what about other logics? What about membership equational logic itself? What about many-sorted equational logic? What about Horn logic with equality? We have for a long time conjectured that these logics are also reflective, and that the same methods developed for rewriting logic can be used to obtain reflection theorems for these new logics. The present work establishes the truth of these conjectures. Furthermore, our constructions shed light on the question of how the universal theories of related logics are themselves related. For example, membership equational logic is itself a sublogic of rewriting logic, and this sublogic relation is expressed at the reflective level by the fact that the universal theory of membership equational logic is itself a *subtheory* of the universal theory for the more general version of rewriting logic where the underlying equational specifications are membership equational theories.

Therefore, our results make clear that reflection is available as a very powerful feature not only for this more general variant of rewriting logic, but also for other computational logics of great importance in formal specification and declarative programming, such as membership equational logic, many-sorted equational logic, and Horn logic with equality. This can then serve as a basis for the theoretically grounded design of *declarative reflective programming languages* in those logics.

In particular, this work provides solid foundations for many useful applications of reflection in Maude [11–13], which implements the most general variant of rewriting logic. As explained in [14], reflection and the flexible uses of rewriting logic as a logical and semantic framework [36] have been fruitfully exploited by a large number of authors to develop in Maude a wide range of applications, which include, among many others (see [37]):

- Full Maude [22,23], an extension of Maude written in Maude itself, endowing the language with a very expressive *module algebra* of parameterized modules and module composition with important extensions to support object-oriented modules.
- Internal strategies to guide the rewrite engine in the application of rules [38,24].
- A tool to automatically check an abstract interpretation against user-given properties [26].
- A Church–Rosser checker that analyzes order-sorted equational specifications in Maude to check whether they satisfy the Church–Rosser property [25].
- Real-Time Maude [44], an execution and analysis environment for the specification of real-time and hybrid systems based on a notion of real-time rewrite theories that has a straightforward transformation into an ordinary rewrite theory.
- An inductive theorem prover [18] that can be used to prove inductive properties of membership equational specifications in Maude. This tool can be extended with reflective reasoning principles to reason about the metalogical properties of a logic represented in rewriting logic [3,15].
- A proof assistant built by Stehr for the Open Calculus of Constructions, which extends Coquand and Huet's calculus of constructions [49].
- Executable specifications of models of computation [2,52,50,9].

The paper is organized as follows. First, in Section 2, we summarize the axioms characterizing the notion of a reflective logic. Then, in Sections 3, 4, 5 and 6, we prove, respectively, that membership equational logic, many-

sorted equational logic, many-sorted Horn logic with equality, and rewriting logic are reflective in our axiomatic sense. Finally, in Section 7 we compare these results with previous work, and in Section 8 we draw conclusions.

2. Reflection in general logics

Although we have studied and exploited the concept of reflection mostly in the context of rewriting logic, it is certainly not specific to this logic. Indeed, the concept of reflection can be axiomatized in a way that applies to arbitrary logics [16].

We present below in summarized form the axiom characterizing the notion of a *reflective logic*. We introduce first the notions of *syntax* and of *entailment system*, used in our axiomatization. These notions are defined using the language of category theory, but do not require any acquaintance with categories beyond the basic notions of category and functor.

Syntax. Syntax can typically be given by a *signature* Σ providing a grammar on which to build *sentences*. For first-order logic, for example, a typical signature consists of a set of function symbols and a set of predicate symbols, each with a prescribed number of arguments, which are used to build up the usual sentences. We assume that for each logic there is a category **Sign** of possible signatures for it, and a functor $sen : \mathbf{Sign} \longrightarrow \mathbf{Set}$ assigning to each signature Σ the set $sen(\Sigma)$ of all its sentences. We call the pair (**Sign**, *sen*) a *syntax*.

Entailment systems. For a given signature Σ in **Sign**, *entailment* (also called *provability*) of a sentence $\varphi \in sen(\Sigma)$ from a set of axioms $\Gamma \subseteq sen(\Sigma)$ is a relation $\Gamma \vdash_\Sigma \varphi$ which holds if and only if we can prove φ from the axioms Γ using the rules of the logic. We make this relation relative to a signature.

In what follows, $|\mathcal{C}|$ denotes the collection of objects of a category $\mathcal{C}$.

Definition 1 (*[40]*). An *entailment system* is a triple $\mathcal{E} = (\mathbf{Sign}, sen, \vdash)$ such that

- (**Sign**, *sen*) is a syntax,
- $\vdash$ is a function associating to each $\Sigma \in |\mathbf{Sign}|$ a binary relation $\vdash_\Sigma \subseteq \mathcal{P}(sen(\Sigma)) \times sen(\Sigma)$, called Σ-*entailment*, that satisfies the following properties:
  1. *reflexivity*: for any $\varphi \in sen(\Sigma)$, $\{\varphi\} \vdash_\Sigma \varphi$,
  2. *monotonicity*: if $\Gamma \vdash_\Sigma \varphi$ and $\Gamma' \supseteq \Gamma$ then $\Gamma' \vdash_\Sigma \varphi$,
  3. *transitivity*: if $\Gamma \vdash_\Sigma \varphi$ for all $\varphi \in \Delta$, and $\Gamma \cup \Delta \vdash_\Sigma \psi$, then $\Gamma \vdash_\Sigma \psi$,
  4. $\vdash$-*translation*: if $\Gamma \vdash_\Sigma \varphi$, then for any $H : \Sigma \to \Sigma'$ in **Sign** we have $sen(H)(\Gamma) \vdash_{\Sigma'} sen(H)(\varphi)$.

The entailment relation $\vdash$ induces a function mapping each set of sentences Γ to the set $\Gamma^\bullet = \{\varphi \mid \Gamma \vdash \varphi\}$. We call $\Gamma^\bullet$ the set of *theorems* provable from Γ.

Definition 2 (*[40]*). Given an entailment system $\mathcal{E}$, its category **Th** of *theories* has as objects pairs $T = (\Sigma, \Gamma)$ with Σ a signature and $\Gamma \subseteq sen(\Sigma)$. A *theory morphism* $H : (\Sigma, \Gamma) \to (\Sigma', \Gamma')$ is a signature morphism $H : \Sigma \to \Sigma'$ such that if $\varphi \in \Gamma$, then $\Gamma' \vdash_{\Sigma'} sen(H)(\varphi)$.

Note that we can extend the functor *sen* to a functor $sen : \mathbf{Th} \longrightarrow \mathbf{Set}$ by taking $sen(\Sigma, \Gamma) = sen(\Sigma)$.

2.1. Reflective logics

A reflective logic is a logic in which important aspects of its metatheory can be represented at the object level in a consistent way, so that the object-level representation correctly simulates the relevant metatheoretic aspects. Two obvious metatheoretic notions that can be so reflected are theories and the entailment relation $\vdash$. This leads us to the notion of a universal theory and, more generally, to the notion of a universal theory relative to a class $\mathcal{C}$ of *representable* theories introduced in Definition 3 below.

Typically, for a theory to be representable at the object level it must have a finitary description in some way—say, being recursively enumerable—so that it can be represented as a piece of language. In the terminology of Shoenfield's axiomatic approach to computability [47], we should require that theories T in $\mathcal{C}$ are *finite objects*; that is, in Shoenfield's own words, "object(s) which can be specified by a finite amount of information"; whereas the

class of theories $\mathcal{C}$ should be a *space*, that is, a class X of finite objects such that, given a finite object x, we can decide whether or not x belongs to X. Computer scientists typically call such finite objects *data structures*, and such spaces *data types*. Of course, in typical finitary logics, if the signature of a theory T is a finite object, then the set $sen(T)$ of its sentences is also a space in the above sense; that is, such sentences are finite objects, and we can effectively determine when they are legal sentences in T. Given spaces X and Y, Shoenfield's notion of a *recursive function* $f : X \longrightarrow Y$ is then a (total) function that can be computed by an *algorithm*, i.e., by a computer program when we disregard space and time limitations.

Definition 3. Given an entailment system $\mathcal{E}$ and a set of theories $\mathcal{C} \subseteq |\mathbf{Th}|$, a theory U is $\mathcal{C}$*-universal* if there is a function, called a *representation function*,

$$\overline{\_ \vdash \_} : \bigcup_{T \in \mathcal{C}} \{T\} \times sen(T) \longrightarrow sen(U)$$

such that for each $T \in \mathcal{C}$, $\varphi \in sen(T)$,

$$T \vdash \varphi \iff U \vdash \overline{T \vdash \varphi}.$$

If, in addition, $U \in \mathcal{C}$, then the entailment system $\mathcal{E}$ is called $\mathcal{C}$*-reflective*.

To take into account computability considerations, we should further require that the representation function $\overline{\_ \vdash \_}$ is *recursive*.[1] Finally, to rule out unfaithful representations, we should require that the function $\overline{\_ \vdash \_}$ is *injective*.[2]

Note that in a reflective entailment system, since U itself is representable, representation can be iterated so that we immediately have a "reflective tower":

$$T \vdash \varphi \iff U \vdash \overline{T \vdash \varphi} \iff U \vdash \overline{U \vdash \overline{T \vdash \varphi}} \ \ldots$$

3. Reflection in membership equational logic

Membership equational logic—in short, MEL—is an expressive version of equational logic supporting sorts, subsorts, operator overloading, and partiality. A full account of its syntax and semantics can be found in [5,43]; here we define the basic notions needed in this paper.

A *signature* in MEL is a triple $\Omega = (K, \Sigma, S)$, with K a set of *kinds*, Σ a K-kinded signature $\Sigma = \{\Sigma_{w,k}\}_{(w,k) \in K^* \times K}$, and $S = \{S_k\}_{k \in K}$ a pairwise disjoint K-kinded family of sets. We call S_k the set of *sorts* of kind k. The pair (K, Σ) is what is usually called a many-sorted signature of function symbols; however we call the elements of K kinds because each kind k now has a set S_k of associated *sorts*, which in the models will be interpreted as subsets of the carrier for the kind. The kind of a sort s is denoted by $[s]$. As usual, we denote by T_Σ the K-kinded algebra of ground Σ-terms, and by $T_\Sigma(X)$ the K-kinded algebra of Σ-terms on the K-kinded set of variables X.

The atomic formulae of MEL are either *equations* $t = t'$, where t and t' are Σ-terms of the same kind, or *membership assertions* of the form $t : s$, where the term t has kind k and $s \in S_k$. Sentences are Horn clauses on these atomic formulae, i.e., sentences of the form

$$(\forall X)\ A_0 \ \textbf{if}\ A_1 \wedge \cdots \wedge A_n,$$

where each A_i is either an equation or a membership assertion, and X is a K-kinded set of variables. A theory in membership equational logic is a pair (Ω, E), where E is a set of sentences—(conditional) equations or (conditional) membership axioms—in MEL over the signature Ω. We refer to [5,43] for the semantics notions of Ω-algebra, and initial and free models.

To simplify the definition of the universal theory for MEL in Section 3.1 we will work with theories with *nonempty kinds*, that is, for each kind, the elements of that kind in the initial algebra form a nonempty set. This is a

[1] Note that, under the assumptions mentioned before, $\bigcup_{T \in \mathcal{C}} \{T\} \times sen(T)$ is a space, since its elements must be pairs of finite elements of the form (T, φ), where we can first decide if T is in the space $\mathcal{C}$, and then decide whether φ is in the space $sen(T)$.

[2] If the entailment relation were decidable, $\overline{T \vdash \varphi}$ could simply be defined to be a sentence provable in U if $T \vdash \varphi$, or an unprovable sentence otherwise; injectivity is required to avoid such pathological definitions. Actually injectivity may not be enough and, although we won't further pursue this issue, one could think of a stronger codification with separate encodings for theories and sentences, which is in fact what we do in our study of rewriting logic.

relatively minor restriction that avoids the well-known complications with quantification in many-sorted equational deduction [31].[3] Thus, from now on, we can omit the quantifiers in all sentences. Also, in what follows we will only deal with *finitely presentable* theories in MEL.

Before we present the rules of deduction for MEL, we need to introduce our notions for contexts and substitutions. Given a signature $\Omega = (K, \Sigma, S)$, a K-kinded set of variables X, and a K-kinded set of new constants $\{\iota_k\}_{k\in K}$, a *context* is a term C^k, $k \in K$, which contains exactly one subterm ι_k, called its "hole". We define $\mathrm{C}^{\iota}_{\Sigma}(X)$ to be the set of contexts. Given a context C^k and a term $t \in T_\Sigma(X)$ of kind k, $C^k[t] \in T_\Sigma(X)$ is the term that results from replacing the "hole" ι_k in C^k by t. When not needed, we omit mentioning the kind of the "hole" in the context. Given a signature $\Omega = (K, \Sigma, S)$ and a set of variables $X = \{x_1, \ldots, x_n\}$, we define $\mathcal{S}_{(\Sigma,X)}$ to be the set of *substitutions*[4]

$$\{(x_1 \mapsto w_1, \ldots, x_n \mapsto w_n) \mid x_i \neq x_j \text{ if } i \neq j,\ w_i \in T_\Sigma(X), \text{ and } x_i \text{ and } w_i \text{ have the same kind}\}.$$

Given a term t and a substitution $\sigma = (x_1 \mapsto w_1, \ldots, x_n \mapsto w_n)$, we denote by $\sigma(t)$ the term $t(w_1/x_1, \ldots, w_n/x_n)$ obtained from t by simultaneously substituting w_i for x_i, $i = 1, \ldots, n$.

The rules of MEL**.** We now introduce the rules of deduction of MEL. Our formulation is slightly different from that in [43], but equivalent to it and simpler for our purposes, in that the congruence rule is removed and is taken into account as part of the (**Replacement**) rule.

Given a membership equational theory $T = (\Omega, E)$, we say that T *entails* a sentence ϕ if and only if $T \vdash \phi$ can be obtained by finite application of the following *rules of deduction*. If T is clear from the context we will simply say that there exists a derivation of ϕ.

1. **Reflexivity**. For every $t \in T_\Sigma(X)$,

$$\frac{}{T \vdash t = t}.$$

2. **Replacement**. For each equation $t = t'$ **if** $C_{mb} \wedge C_{eq}$ in E, with t, t' of kind k, context $C^k \in \mathrm{C}^{\iota}_{\Sigma}(X)$, and substitution σ, where $C_{mb} \triangleq (u_1 : s_1 \wedge \cdots \wedge u_j : s_j)$ and $C_{eq} \triangleq (v_1 = v'_1 \wedge \cdots \wedge v_k = v'_k)$,

$$\frac{T \vdash \sigma(u_1) : s_1 \quad \cdots \quad T \vdash \sigma(u_j) : s_j \quad T \vdash \sigma(v_1) = \sigma(v'_1) \quad \cdots \quad T \vdash \sigma(v_k) = \sigma(v'_k)}{T \vdash C^k[\sigma(t)] = C^k[\sigma(t')]}.$$

   Similarly, for each membership axiom $t : s$ **if** $C_{mb} \wedge C_{eq}$ in E, and substitution σ, where C_{mb} and C_{eq} are as before,

$$\frac{T \vdash \sigma(u_1) : s_1 \quad \cdots \quad T \vdash \sigma(u_j) : s_j \quad T \vdash \sigma(v_1) = \sigma(v'_1) \quad \cdots \quad T \vdash \sigma(v_k) = \sigma(v'_k)}{T \vdash \sigma(t) : s}.$$

3. **Symmetry**. For every $t, t' \in T_\Sigma(X)$,

$$\frac{T \vdash t = t'}{T \vdash t' = t}.$$

4. **Transitivity**. For every $t, t', t'' \in T_\Sigma(X)$,

$$\frac{T \vdash t = t'' \qquad T \vdash t'' = t'}{T \vdash t = t'}.$$

5. **Membership**. For every $t, u \in T_\Sigma(X)$,

$$\frac{T \vdash t = u \qquad T \vdash u : s}{T \vdash t : s}.$$

[3] The specification and proof of correctness of a universal theory in the presence of empty kinds follows very similar lines. The main difference is that the universal quantifiers need to be metarepresented in sentences and the inference rules must keep track of such quantifiers.

[4] Conceptually, a substitution is a function from variables to terms. For technical convenience, we choose to define substitutions as a special case of lists of pairs formed by variables and terms.

3.1. A universal theory for MEL

In this section we introduce the universal theory U_{MEL} and a representation function $\overline{\_ \vdash \_}$ that encodes pairs consisting of a theory and a sentence over its signature as a sentence in U_{MEL}.

The signature of U_{MEL}. The signature of the theory U_{MEL} contains operators to represent terms, contexts, substitutions, kinds, sorts, signatures, axioms, and theories. To ease readability we present the signature of U_{MEL} as a Maude specification, using "mixfix" syntax. For example, the conjunction operator is specified with infix syntax `_and_` where the underbars indicate the place of the two arguments. In what follows we enumerate the operators in the signature of U_{MEL}; its kinds are those that appear in the operator declarations and no sorts are used.

The main operators in U_{MEL} are the following. First of all, the signature of U_{MEL} contains a small subset of the basic Boolean operators.

```
op true : -> [Bool] .
op false : -> [Bool] .
op _and_ : [Bool] [Bool] -> [Bool] .
op _or_ : [Bool] [Bool] -> [Bool] .
```

Lists of ASCII characters preceded by a quote, built with the operators `nilM` and `consM`, are used to represent the symbols of a MEL theory. There are also two equality predicates that return `true` when two such characters or lists are syntactically equal, and `false` otherwise.

```
ops 'a 'b 'c ... : -> [ASCII] .
op equalASCII : [ASCII] [ASCII] -> [Bool] .
op nilM : -> [MetaExp] .
op consM : [ASCII] [MetaExp] -> [MetaExp] .
op equalM : [MetaExp] [MetaExp] -> [Bool] .
```

To represent kinds, lists of kinds, and an equality predicate for kinds we use the following operators:

```
op k : [MetaExp] -> [Kind] .
op nilK : -> [KindList] .
op consK : [Kind] [KindList] -> [KindList] .
op equalK : [Kind] [Kind] -> [Bool] .
```

Variables, arbitrary terms, and lists of terms are represented using

```
op v : [MetaExp] [Kind]  -> [Term] .
op _[_] : [MetaExp] [TermList] -> [Term] .
op nilTL : -> [TermList] .
op consTL : [Term] [TermList] -> [TermList] .
```

Note that the kind of a variable is metarepresented together with its name. During the rest of the proof, since the metarepresentation always needs both the name x and the kind k, we will also refer to variables at the object level as pairs $x : k$.

For example, the term $x : [Nat] + (y : [Nat] - 0)$ is metarepresented in U_{MEL} as

```
consM('_,consM('+,consM('_,nilM)))
    [consTL(v('x,k(consM('N,consM('a,consM('t,nilM))))),
            consTL(consM('_,consM('-,consM('_,nilM)))
                       [consTL(v('y,k(consM('N,consM('a,consM('t,nilM))))),
                               consTL('0[nilTL],nilTL))],nilTL))]
```

Sorts and operators in a signature are represented using the following operators:

```
op s : [MetaExp] [Kind] -> [Sort] .
op _:_->_ : [MetaExp] [KindList] [Kind] -> [Operator] .
```

Substitutions are represented with the following operators, where '-' represents the empty substitution that acts as the identity over terms.

```
op - : -> [Substitution] .
op v(_,_)->_ : [MetaExp] [Kind] [Term] -> [Assignment] .
op consS : [Assignment] [Substitution] -> [Substitution] .
```

For contexts, we have the following operators:

```
op * : -> [Context] .
op _[_] : [MetaExp] [CTermList] -> [Context] .
op consCTL1 : [Context] [TermList] -> [CTermList] .
op consCTL2 : [Term] [CTermList] -> [CTermList] .
```

The kind `[CTermList]` is introduced to represent lists of terms such that there is only one "hole" among them, that is, only one of the terms is a context. Due to this, there exist two different `cons` operators to build these lists: one to append arbitrary terms to the left and another one to append a context when there is none yet. For example, the context $0 + (\iota - x : [Nat])$ is metarepresented as

```
consM('_,consM('+,consM('_,nilM)))
    [consCTL2('0[nilTL],
              consCTL1(consM('_,consM('-,consM('_,nilM)))
                       [consCTL1(*,
                          consTL(v('x,k(consM('N,consM('a,consM('t,nilM))))),
                          nilTL))],
                    nilTL))]
```

To represent (possibly conditional) equations and membership axioms, the signature of U_{MEL} includes the operators

```
op _=_ : [Term] [Term] -> [Atom] .
op _:_ : [Term] [Sort] -> [Atom] .
op none : -> [Condition] .
op _/\_ : [Atom] [Condition] -> [Condition] .
op _if_ : [Atom] [Condition] -> [Axiom] .
```

where the constant `none` is used to represent the lack of conditions. The operators

```
op (_,_,_) : [KindSet] [OperatorSet] [SortSet] -> [Signature] .
op (_,_) : [Signature] [AxiomSet] -> [MelTheory] .
```

are the ones used to represent signatures and theories, respectively. Finally, the signature of U_{MEL} contains the Boolean operator

```
op _|-_ : [MelTheory] [Axiom] -> [Bool] .
```

to represent entailment of sentences in a given membership equational theory; the main axioms of U_{MEL}, including those in Fig. 1, define the behavior of this operator.

In addition, the signature of U_{MEL} contains operators for sets of kinds, axioms, sorts, and operators.

```
op emptyK : -> [KindSet] .
op unionK : [Kind] [KindSet] -> [KindSet] .
op emptyA : -> [AxiomSet] .
op unionA : [Axiom] [AxiomSet] -> [AxiomSet] .
op emptyS : -> [SortSet] .
op unionS : [Sort] [SortSet] -> [SortSet] .
op emptyOp -> [OperatorSet] .
op unionOp : [Operator] [OperatorSet] -> [OperatorSet] .
```

In the signature of U_{MEL} we also have some Boolean operators `parse` to decide whether a term is well-formed with respect to a many-kinded signature, and operators `applyC` and `applyS` to apply a context and a substitution respectively to a term.

```
op parse : [Term] [Signature] -> [Bool] .
op parse : [Term] [Signature] [Kind] -> [Bool] .
op parse : [TermList] [Signature] [KindList] -> [Bool] .
op applyC : [Context] [Term] -> [Term] .
```

```
op applyC : [CTermList] [Term] -> [TermList] .
op applyS : [Substitution] [Term] -> [Term] .
op applyS : [Substitution] [TermList] -> [TermList] .
```

Finally, the operators

```
op satisfyC : [MelTheory] [Condition] [Substitution] -> [Bool] .
op satisfyA : [MelTheory] [Atom] [Substitution] -> [Bool] .
```

are used to decide, given a theory T, a condition C, and a substitution σ, whether all atomic formulae in $\sigma(C)$ can be proved in T.

The representation function. We next define the representation function $\overline{\_ \vdash \_}$. For all membership equational theories T and sentences ϕ over the signature of T,

$$\overline{T \vdash \phi} \triangleq (\overline{T} \,\texttt{|-}\, \overline{\phi}) = \texttt{true},$$

where $\overline{(\_)}$ is a representation function defined recursively in the following way:

1. For a theory $T = (\Omega, E)$,
   $\overline{T} \triangleq (\overline{\Omega}, \overline{E})$.
2. For E a set of sentences $\{\varphi_1, \ldots, \varphi_n\}$,
   $\overline{E} \triangleq \texttt{unionA(}\overline{\varphi_1}\texttt{,} \ldots \texttt{,unionA(}\overline{\varphi_n}\texttt{,emptyA)} \ldots \texttt{)}$.
3. For atomic formulae φ and ψ of the form $t = t'$ and $t : s$ respectively,
   $\overline{\varphi} \triangleq \overline{t} = \overline{t'}$ and $\overline{\psi} \triangleq \overline{t} : \overline{s}$.
4. For *Cond* a conjunction of atomic formulae $A_1 \wedge A_2 \wedge \cdots \wedge A_n$,
   $\overline{Cond} \triangleq \overline{A_1}\ \texttt{/\textbackslash}\ (\overline{A_2}\ \texttt{/\textbackslash}\ \cdots\ \texttt{/\textbackslash}\ (\overline{A_n}\ \texttt{/\textbackslash}\ \texttt{none}) \ldots)$.
5. For a sentence $\varphi = A_0$ **if** $A_1 \wedge \cdots \wedge A_n$,
   $\overline{\varphi} \triangleq \overline{A_0}\ \texttt{if}\ \overline{A_1 \wedge \cdots \wedge A_n}$.
6. For a l a nonempty string $(a_1, \ldots, a_n)$ of ASCII characters,
   $\overline{l} \triangleq \texttt{consM('}a_1\texttt{,} \ldots \texttt{,consM('}a_n\texttt{,nilM)} \ldots \texttt{)}$.
7. For a variable x of kind k,
   $\overline{x : k} \triangleq \texttt{v(}\overline{x}\texttt{,}\overline{k}\texttt{)}$.
8. For t a term of the form $f(t_1, \ldots, t_n)$,
   $\overline{t} \triangleq \overline{f}[\overline{t_1, \ldots, t_n}]$.
9. For tl a list of terms $(t_1, \ldots, t_n)$,
   $\overline{tl} \triangleq \texttt{consTL(}\overline{t_1}\texttt{,} \ldots \texttt{,consTL(}\overline{t_n}\texttt{,nilTL)} \ldots \texttt{)}$.
10. For σ a substitution $x_1 : k_1 \mapsto w_1, \ldots, x_n : k_n \mapsto w_n$,
    $\overline{\sigma} \triangleq \texttt{consS(v(}\overline{x_1}\texttt{,}\overline{k_1}\texttt{)\ ->\ }\overline{w_1}\texttt{,...,consS(v(}\overline{x_n}\texttt{,\ }\overline{k_n}\texttt{)\ ->\ }\overline{w_n}\texttt{,-)...)}$.

And analogously for the rest of the elements: contexts, signatures, and sets of kinds, sorts and operators.

Notice that, as they stand, some of the definitions are ambiguous. For example, in item 2, depending on how we order the elements in the set we may get different metarepresentations. This problem will disappear in the next section after we introduce appropriate equations for all the set operators.

The axioms of $\mathrm{U}_{\mathrm{MEL}}$**.** We now define the axioms of $\mathrm{U}_{\mathrm{MEL}}$, which include equations that correspond to the inference rules of MEL, along with equations that define the previously presented operators: `parse`, `applyC`, `satisfyC`, ... To ease the understanding of these equations, we replace the usual variable notation by the corresponding representations of the entities to be placed in such variable positions. For example, $\overline{\Omega}$ is a normal variable, but the notation suggests that the terms that the variable will match will typically be representations of signatures.

We start by giving the obvious equations for the Boolean operators:

```
eq true and B = B .
eq false and B = false .
eq true or B = true .
eq false or B = B .
```

Next, the equations for the set operators; for sets of axioms:

```
eq unionA(φ, unionA(ψ,AS)) = unionA(ψ, unionA(φ,AS)) .
eq unionA(φ, unionA(φ,AS)) = unionA(φ,AS) .
```

and analogously for sets of kinds, sorts, and operators, and for the `_/\_` operator that builds conditions:

```
eq φ /\ (ψ /\ Cond) = ψ /\ (φ /\ Cond) .
eq φ /\ (φ /\ Cond) = φ /\ Cond .
```

In particular, we have the following result and (2) above becomes unambiguous.

Proposition 1. *For every set E of sentences and $\varphi \in E$,*

$$\mathrm{U}_{\mathrm{MEL}} \vdash \overline{E} = \texttt{unionA}(\overline{\varphi}, \overline{E \setminus \{\varphi\}}).$$

Throughout this section, several auxiliary results of this form will be stated. Their proofs are usually straightforward, but tedious, inductions and therefore will be omitted unless they require an additional explanation.

To define the equality predicate for ASCII characters we just have to consider all possibilities:

```
eq equalASCII('a, 'a) = true .
eq equalASCII('a, 'b) = false .
eq equalASCII('a, 'c) = false .
...
```

and its extension to lists of characters is straightforward:

```
eq equalM(nilM, nilM) = true .
eq equalM(nilM, consM(c,cl)) = false .
eq equalM(consM(c,cl), nilM) = false .
eq equalM(consM(c,cl), consM(c',cl')) = equalASCII(c,c') and equalM(cl,cl') .
```

The specification of the equality predicate for kinds is now immediate:

```
eq equalK(k(cl), k(cl')) = equalM(cl,cl') .
```

Then it can be proved by structural induction on terms that:

Proposition 2. *For all terms w, w' in $\mathrm{U}_{\mathrm{MEL}}$ of kind, respectively,* [ASCII], [MetaExp], *and* [Kind], *and for f equal to, respectively,* equalASCII, equalM, *and* equalK*:*

- *w and w' are syntactically equal iff $\mathrm{U}_{\mathrm{MEL}} \vdash f(w, w') =$* true*, and*
- *w and w' are syntactically different iff $\mathrm{U}_{\mathrm{MEL}} \vdash f(w, w') =$* false*.*

Proof. It is enough to note that the operators are free (no equations have been given, nor will be given, between terms of these kinds) and that the equality predicates only identify terms that have been constructed in the same way. □

Next we introduce the equations that define the applications of contexts and substitutions to terms. We first treat contexts.

```
eq applyC(*,t) = t .
eq applyC(f[ctl], t) = f[applyC(ctl,t)] .

eq applyC(consCTL1(C,tl), t) = consTL(applyC(C,t),tl) .
eq applyC(consCTL2(t',ctl), t) = consTL(t',applyC(ctl,t)) .
```

Proposition 3. *For all terms $t \in T_\Sigma(X)$ and contexts $C \in \mathrm{C}^{\iota}_{\Sigma}$, it holds that*

$$\mathrm{U}_{\mathrm{MEL}} \vdash \texttt{applyC}(\overline{C}, \overline{t}) = \overline{C[t]}.$$

Similarly, we give the equations that define the application of a substitution to a given term.

```
eq applyS(-,t̄) = t̄ .
eq applyS(consS(v(x̄,k̄)-> w̄,σ̄), v(x̄,k̄)) = w̄ .
ceq applyS(consS(v(x̄′,k̄′)-> w̄,σ̄), v(x̄,k̄)) = applyS(σ̄, v(x̄,k̄))
    if equalM(x̄,x̄′) and equalK(k̄,k̄′) = false .
eq applyS(σ̄,f̄[t̄l̄]) = f̄[applyS(σ̄,t̄l̄)] .

eq applyS(σ̄, nilTL) = nilTL .
eq applyS(σ̄, consTL(t̄,t̄l̄)) = consTL(applyS(σ̄,t̄), applyS(σ̄,t̄l̄)) .
```

Proposition 4. *For all terms $t \in T_\Sigma(X)$ and substitutions σ, it holds that*

$$\mathrm{U}_{\mathrm{MEL}} \vdash \texttt{applyS}(\overline{\sigma},\overline{t}) = \overline{\sigma(t)}.$$

Since the representation function is injective and `applyC` and `applyS` are the only operators, besides the constructors, that return terms of kind `[Term]`, the next result is clear.

Proposition 5. *For all terms $t_1, t_2 \in T_\Sigma(X)$, if $\mathrm{U}_{\mathrm{MEL}} \vdash \overline{t_1} = \overline{t_2}$ then t_1 is syntactically equal to t_2.*

To check if a term is well-formed we introduce the following equations:

```
ceq parse(t̄,(K̄,Σ̄,S̄)) = true
    if K̄ = unionK(k̄,K̄) /\ parse(t̄, (K̄,Σ̄,S̄),k̄) = true .
eq parse(v(x̄,k̄), Ω̄, k̄) = true .
ceq parse(f̄[t̄l̄], (K̄,Σ̄,S̄), k̄) = true
    if Σ̄ = unionOp(f̄ : k̄l̄ -> k̄, Σ̄) /\ parse(t̄l̄, (K̄,Σ̄,S̄), k̄l̄) = true .
eq parse(nilTL, Ω̄, nilK) = true .
ceq parse(consTL(t̄, t̄l̄), Ω̄, consK(k̄, k̄l̄)) = true
    if parse(t̄, Ω̄, k̄) = true /\ parse(t̄l̄, Ω̄, k̄) = true .
```

The following proposition is then a straightforward consequence.

Proposition 6. *For all terms $t \in T_\Sigma(X)$,*

$$\mathrm{U}_{\mathrm{MEL}} \vdash \texttt{parse}(\overline{t},\overline{\Omega}) = \texttt{true}.$$

Furthermore, for all terms w in $\mathrm{U}_{\mathrm{MEL}}$ of kind `[Term]`*, it holds that if*

$$\mathrm{U}_{\mathrm{MEL}} \vdash \texttt{parse}(w,\overline{\Omega}) = \texttt{true},$$

then there is a term $t \in T_\Sigma(X)$ such that $\mathrm{U}_{\mathrm{MEL}} \vdash w = \overline{t}$.

Proof. The first part is immediate by structural induction on t. For the second part, which is much more tedious and has to carefully consider the derivations that arise from using **(Transitivity)**, note that we cannot conclude that w is $\overline{t}$ due to the operators `applyC` and `applyS`. □

The auxiliary operator `satisfyC` checks whether a condition holds in a theory under a given substitution. The equations simply iterate through all the atoms in the condition and leave the hard work of checking whether they are satisfied to (_|-_).

```
eq satisfyC(T̄,none,σ̄) = true .
eq satisfyC(T̄,A̅ /\ C̄,σ̄) = satisfyA(T̄,A̅,σ̄) and satisfyC(T̄,C̄,σ̄) .

eq satisfyA(T̄,t̄ = t̄′,σ̄) = (T̄ |- applyS(σ̄,t̄) = applyS(σ̄,t̄′) if none) .
eq satisfyA(T̄,t̄ : s̄,σ̄) = (T̄ |- applyS(σ̄,t̄) : s̄ if none) .
```

Proposition 7. *For all atomic formulae $A_1, \ldots, A_n$ and substitutions σ, the following are equivalent:*

1. $\mathrm{U}_{\mathrm{MEL}} \vdash \overline{T \vdash \sigma(A_i)}$ *for each i; that is, for each A_i of the form $t_i = t_i'$,*

$$\mathrm{U}_{\mathrm{MEL}} \vdash (\overline{T} \texttt{ |- } \overline{\sigma(t_i)} = \overline{\sigma(t_i')} \texttt{ if none}) = \texttt{true},$$

and for each A_i of the form $t_i : s_i$,

$$\mathrm{U}_{\mathrm{MEL}} \vdash (\overline{T} \ \texttt{|-}\ \overline{\sigma(t_i)} : \overline{s_i} \ \texttt{if none}) = \texttt{true}.$$

2. *It holds that*

$$\mathrm{U}_{\mathrm{MEL}} \vdash \texttt{satisfyC}(\overline{T}, \overline{A_1 \wedge \cdots \wedge A_n}, \overline{\sigma}) = \texttt{true}.$$

Proof. It can be easily shown by induction on n that (2) is equivalent to: for each A_i of the form $t_i = t_i'$,

$$\mathrm{U}_{\mathrm{MEL}} \vdash (\overline{T} \ \texttt{|-}\ \texttt{applyS}(\overline{\sigma}, \overline{t_i}) = \texttt{applyS}(\overline{\sigma}, \overline{t_i}') \ \texttt{if none}) = \texttt{true},$$

and for each A_i of the form $t_i : s_i$,

$$\mathrm{U}_{\mathrm{MEL}} \vdash (\overline{T} \ \texttt{|-}\ \texttt{applyS}(\overline{\sigma}, \overline{t_i}) : \overline{s_i} \ \texttt{if none}) = \texttt{true}.$$

The result now follows from Proposition 4. □

The specification in $\mathrm{U}_{\mathrm{MEL}}$ of the inference rules of MEL is now immediate and they appear listed in Fig. 1.

```
*** reflexivity
eq ((Ω̄,Ē) |- t̄ = t̄ if none) = true .

*** replacement
ceq ((Ω̄,Ē) |- applyC(C̄, applyS(σ̄,t̄)) = applyC(C̄, applyS(σ̄,t̄')) if none) = true
    if Ē = unionA(t̄ = t̄' if Cond,E')
    /\ satisfyC((Ω̄,Ē), Cond, σ̄) = true .

*** replacement
ceq ((Ω̄,Ē) |- applyS(σ̄,t̄) : s̄ if none) = true
    if Ē = unionA(t̄ : s̄ if Cond,E')
    /\ satisfyC((Ω̄,Ē), Cond, σ̄) = true .

*** symmetry
ceq ((Ω̄,Ē) |- t̄ = t̄' if none) = true
    if ((Ω̄,Ē) |- t̄' = t̄ if none) = true .

*** transitivity
ceq ((Ω̄,Ē) |- t̄ = t̄' if none) = true
    if parse(t̄'', Ω̄) = true
    /\ ((Ω̄,Ē) |- t̄ = t̄'' if none) = true
    /\ ((Ω̄,Ē) |- t̄'' = t̄' if none) = true .

*** membership
ceq ((Ω̄, Ē) |- t̄ : s̄ if none) = true
    if parse(ū, Ω̄) = true
    /\ ((Ω̄,Ē) |- t̄ = ū if none) = true
    /\ ((Ω̄,Ē)|- ū : s̄ if none) = true .
```

Fig. 1. The universal theory $\mathrm{U}_{\mathrm{MEL}}$ (fragment).

3.2. *The correctness of the universal theory* $\mathrm{U}_{\mathrm{MEL}}$

The theory $\mathrm{U}_{\mathrm{MEL}}$ presented in the previous sections is universal in the precise sense of Definition 3.

Theorem 1. *For all terms $t \in T_\Sigma(X)$ and sorts s in the signature Ω of a theory T,*

$$T \vdash t : s \iff \mathrm{U}_{\mathrm{MEL}} \vdash \overline{T \vdash t : s}.$$

Similarly, for all $t, t' \in T_\Sigma(X)$ over the signature Ω of T,

$$T \vdash t = t' \iff \mathrm{U}_{\mathrm{MEL}} \vdash \overline{T \vdash t = t'}.$$

Proof. The ($\Rightarrow$)-direction corresponds to Theorem 2 below, whereas the implication in the other direction follows from Theorem 3. □

Let us start with the ($\Rightarrow$)-direction of the theorem.

Theorem 2. *For all terms $t, t' \in T_\Sigma(X)$, and sorts s in Ω,*

$$T \vdash t = t' \implies \mathrm{U}_{\mathrm{MEL}} \vdash (\overline{T} \mid\!- \overline{t} = \overline{t'} \texttt{ if none}) = \texttt{true}$$

and

$$T \vdash t : s \implies \mathrm{U}_{\mathrm{MEL}} \vdash (\overline{T} \mid\!- \overline{t} : \overline{s} \texttt{ if none}) = \texttt{true}.$$

Proof. By structural induction on the derivation of $T \vdash t = t'$ or $T \vdash t : s$. According to the last rule of deduction used in reasoning with T, we have:

- **(Reflexivity)**. The result follows by using **(Replacement)** in reasoning with $\mathrm{U}_{\mathrm{MEL}}$, applied to the equation `reflexivity` in Fig. 1.
- **(Symmetry)**. By induction hypothesis, $\mathrm{U}_{\mathrm{MEL}} \vdash (\overline{T} \mid\!- \overline{t'} = \overline{t} \texttt{ if none}) = \texttt{true}$, and we can then use **(Replacement)** applied to the `symmetry` equation.
- **(Transitivity)**. In the case

$$\frac{T \vdash t = t'' \qquad T \vdash t'' = t'}{T \vdash t = t'},$$

  we have, by induction hypothesis, $\mathrm{U}_{\mathrm{MEL}} \vdash (\overline{T} \mid\!- \overline{t} = \overline{t''} \texttt{ if none}) = \texttt{true}$ and $\mathrm{U}_{\mathrm{MEL}} \vdash (\overline{T} \mid\!- \overline{t''} = \overline{t'} \texttt{ if none}) = \texttt{true}$, and by Proposition 6, $\mathrm{U}_{\mathrm{MEL}} \vdash \texttt{parse}(\overline{t''}, \overline{\Omega}) = \texttt{true}$, so we can apply **(Replacement)** to the `transitivity` equation to get the result.
- **(Membership)**. If

$$\frac{T \vdash t = t' \qquad T \vdash t' : s}{T \vdash t : s},$$

  we have, by induction hypothesis, $\mathrm{U}_{\mathrm{MEL}} \vdash (\overline{T} \mid\!- \overline{t} = \overline{t'} \texttt{ if none}) = \texttt{true}$ and $\mathrm{U}_{\mathrm{MEL}} \vdash (\overline{T} \mid\!- \overline{t'} : \overline{s} \texttt{ if none}) = \texttt{true}$, and by Proposition 6, $\mathrm{U}_{\mathrm{MEL}} \vdash \texttt{parse}(\overline{t'}, \overline{\Omega}) = \texttt{true}$, and the result follows applying **(Replacement)** to the `membership` equation.
- **(Replacement)**. Suppose that $t = t'$ **if** $C_{mb} \wedge C_{eq} \in E$, where $C_{mb} \triangleq (u_1 : s_1 \wedge \cdots \wedge u_j : s_j)$ and $C_{eq} \triangleq (v_1 = v'_1 \wedge \cdots \wedge v_k = v'_k)$, and that we have, for some context C and substitution σ,

$$\frac{T \vdash \sigma(u_1) : s_1 \quad \cdots \quad T \vdash \sigma(u_j) : s_j \quad T \vdash \sigma(v_1) = \sigma(v'_1) \quad \cdots \quad T \vdash \sigma(v_k) = \sigma(v'_k)}{T \vdash C[\sigma(t)] = C[\sigma(t')]}.$$

  By induction hypothesis, $\mathrm{U}_{\mathrm{MEL}} \vdash (\overline{T} \mid\!- \overline{\sigma(u_i)} : \overline{s}_i \texttt{ if none}) = \texttt{true}$, for $i = 1, \ldots, j$, and $\mathrm{U}_{\mathrm{MEL}} \vdash (\overline{T} \mid\!- \overline{\sigma(v_i)} = \overline{\sigma(v'_i)} \texttt{ if none}) = \texttt{true}$, for $i = 1, \ldots, k$. Therefore, by Proposition 7 we can use the first `replacement` equation to get

$$\mathrm{U}_{\mathrm{MEL}} \vdash (\overline{T} \mid\!- \texttt{applyC}(\overline{C}, \texttt{applyS}(\overline{\sigma}, \overline{t})) = \texttt{applyC}(\overline{C}, \texttt{applyS}(\overline{\sigma}, \overline{t'})) \texttt{ if none}) = \texttt{true}.$$

  And then the result follows since, by Propositions 3 and 4,

$$\mathrm{U}_{\mathrm{MEL}} \vdash \texttt{applyC}(\overline{C}, \texttt{applyS}(\overline{\sigma}, \overline{t})) = \overline{C[\sigma(t)]}$$

  and

$$\mathrm{U}_{\mathrm{MEL}} \vdash \texttt{applyC}(\overline{C}, \texttt{applyS}(\overline{\sigma}, \overline{t'})) = \overline{C[\sigma(t')]}.$$

  In case the sentence in E that was applied had been of the form $t : s$ **if** $C_{mb} \wedge C_{eq}$, the argument would be the same but now we would use the second `replacement` equation. □

To prove the ($\Leftarrow$)-direction of the main theorem we need the auxiliary Lemma 1. Essentially, it will allow us to work with "normalized" derivations of minimum depth and to discard those spurious ones that arise through the use of the **(Transitivity)** rule. More precisely, note that once derivations for the equations $\overline{T_1 \vdash t_1 = t_1'} = \mathtt{true}$ and $\overline{T_2 \vdash t_2 = t_2'} = \mathtt{true}$ have been obtained, **(Transitivity)** can be used to first get $\overline{T_1 \vdash t_1 = t_1'} = \overline{T_2 \vdash t_2 = t_2'}$ and then to combine them to get an infinite number of uninteresting derivations of arbitrary depth for the same equations; these last derivations are the ones we want to avoid.

Given a proof δ in MEL of an equation or membership assertion, we denote by $depth(\delta)$ its *depth*, defined in the usual way.

Lemma 1. *For all terms t_1, t_2 and atomic formulae $A_1, \ldots, A_n$ over a membership theory T, and terms w_1, w_2 of kind* `[Term]`, *w_3 of kind* `[Condition]`, *and w_4 of kind* `[MelTheory]` *in* $\mathrm{U}_{\mathrm{MEL}}$ *such that* $\mathrm{U}_{\mathrm{MEL}} \vdash \overline{t_1} = w_1$, $\mathrm{U}_{\mathrm{MEL}} \vdash \overline{t_2} = w_2$, $\mathrm{U}_{\mathrm{MEL}} \vdash \overline{A_1 \wedge \cdots \wedge A_n} = w_3$, *and* $\mathrm{U}_{\mathrm{MEL}} \vdash \overline{T} = w_4$, *whenever there is a proof δ in* $\mathrm{U}_{\mathrm{MEL}}$ *of*

$$(w_4 \mid\!- w_1 = w_2 \ \mathtt{if}\ w_3) = m$$

or of

$$m = (w_4 \mid\!- w_1 = w_2 \ \mathtt{if}\ w_3)$$

for some term m of kind `[Bool]` *over the signature of* $\mathrm{U}_{\mathrm{MEL}}$, *one of the following alternatives holds:*

1. *m is $(w_4' \mid\!- w_1' = w_2' \ \mathtt{if}\ w_3')$, for terms w_1', w_2', w_3', and w_4' such that* $\mathrm{U}_{\mathrm{MEL}} \vdash \overline{t_1} = w_1'$, $\mathrm{U}_{\mathrm{MEL}} \vdash \overline{t_2} = w_2'$, $\mathrm{U}_{\mathrm{MEL}} \vdash \overline{A_1 \wedge \cdots \wedge A_n} = w_3'$, *and* $\mathrm{U}_{\mathrm{MEL}} \vdash \overline{T} = w_4'$; *or*
2. *there exists a proof δ' of $(w_4 \mid\!- w_1 = w_2 \ \mathtt{if}\ w_3) = \mathtt{true}$, or of $\mathtt{true} = (w_4 \mid\!- w_1 = w_2 \ \mathtt{if}\ w_3)$, such that $depth(\delta') \leq depth(\delta)$.*

An analogous result holds when $(w_4 \mid\!- w_1 = w_2 \ \mathtt{if}\ w_3)$ is replaced by $(w_4 \mid\!- w_1 : \overline{s} \ \mathtt{if}\ w_3)$.

Proof. By structural induction on the proof δ; we consider the last rule of deduction employed.

- **(Reflexivity)**. Then (1) must hold.
- **(Symmetry)**. The result follows by induction hypothesis.
- **(Membership)**. It is not possible.
- **(Replacement)**. It must be the case that $(w_4 \mid\!- w_1 = w_2 \ \mathtt{if}\ w_3) = m$ has been obtained by means of either:
  - one of the equations corresponding to the rules of deduction in MEL (Fig. 1) using an empty context, in which case m has to be `true` and thus (2) holds taking δ itself as the proof δ', or
  - an equation applied to w_1, w_2, w_3, or w_4 (or to one of their subterms, through an operator like `vars` or `applyS`). Assuming an equation $w_1 = w_1'$ has been used (analogously for the other cases), the context used must have been $(w_4 \mid\!- \iota = w_2 \ \mathtt{if}\ w_3)$ and then m is $(w_4 \mid\!- w_1' = w_2 \ \mathtt{if}\ w_3)$; hence (1) holds.
- **(Transitivity)**. Suppose that δ proves $(w_4 \mid\!- w_1 = w_2 \ \mathtt{if}\ w_3) = m$ (the symmetric case is analogous). There is a term n such that

$$\frac{\mathrm{U}_{\mathrm{MEL}} \vdash (w_4 \mid\!- w_1 = w_2 \ \mathtt{if}\ w_3) = n \qquad \mathrm{U}_{\mathrm{MEL}} \vdash n = m}{\mathrm{U}_{\mathrm{MEL}} \vdash (w_4 \mid\!- w_1 = w_2 \ \mathtt{if}\ w_3) = m}.$$

  Applying the induction hypothesis to the proof δ' of $(w_4 \mid\!- w_1 = w_2 \ \mathtt{if}\ w_3) = n$, we have one of the following possibilities:
  - n is $(w_4' \mid\!- w_1' = w_2' \ \mathtt{if}\ w_3')$, with derivations for $\mathrm{U}_{\mathrm{MEL}} \vdash \overline{t_1} = w_1'$, $\mathrm{U}_{\mathrm{MEL}} \vdash \overline{t_2} = w_2'$, $\mathrm{U}_{\mathrm{MEL}} \vdash \overline{A_1 \wedge \cdots \wedge A_n} = w_3'$, and $\mathrm{U}_{\mathrm{MEL}} \vdash \overline{T} = w_4'$. In this case we apply the induction hypothesis to the proof δ'' of $n = m$, and distinguish the following cases:
    - m is $w_4'' \mid\!- w_1'' = w_2'' \ \mathtt{if}\ w_3''$, with $\mathrm{U}_{\mathrm{MEL}} \vdash \overline{t_1} = w_1''$, $\mathrm{U}_{\mathrm{MEL}} \vdash \overline{t_2} = w_2''$, $\mathrm{U}_{\mathrm{MEL}} \vdash \overline{A_1 \wedge \cdots \wedge A_n} = w_3''$, and $\mathrm{U}_{\mathrm{MEL}} \vdash \overline{T} = w''$, and (1) holds.
    - There is a proof δ''' of $n = \mathtt{true}$ such that $depth(\delta''') \leq depth(\delta'')$. But then
$$\frac{\mathrm{U}_{\mathrm{MEL}} \vdash (w_4 \mid\!- w_1 = w_2 \ \mathtt{if}\ w_3) = n \qquad \mathrm{U}_{\mathrm{MEL}} \vdash n = \mathtt{true}}{(w_4 \mid\!- w_1 = w_2 \ \mathtt{if}\ w_3) = \mathtt{true}}$$
      is a proof of $\mathrm{U}_{\mathrm{MEL}} \vdash (w_4 \mid\!- w_1 = w_2 \ \mathtt{if}\ w_3) = \mathtt{true}$ whose depth is equal to $1 + \max(depth(\delta'), depth(\delta''')) \leq 1 + \max(depth(\delta'), depth(\delta'')) = depth(\delta)$, and we have (2).

- There exists a proof of $(w_4 \mid\!- w_1 = w_2 \texttt{ if } w_3) = \texttt{true}$ whose depth is less than, or equal to, that of δ' and, therefore, less than that of δ, and (2) holds. □

Theorem 3. *Let t_1 and t_2 be terms over a membership theory T, and w_1, w_2, w_3, and w_4 be terms in $\mathrm{U}_{\mathrm{MEL}}$ such that $\mathrm{U}_{\mathrm{MEL}} \vdash \overline{t_1} = w_1$, $\mathrm{U}_{\mathrm{MEL}} \vdash \overline{t_2} = w_2$, $\mathrm{U}_{\mathrm{MEL}} \vdash \texttt{none} = w_3$, and $\mathrm{U}_{\mathrm{MEL}} \vdash \overline{T} = w_4$. If*

$$\mathrm{U}_{\mathrm{MEL}} \vdash (w_4 \mid\!- w_1 = w_2 \texttt{ if } w_3) = \texttt{true}$$

or

$$\mathrm{U}_{\mathrm{MEL}} \vdash \texttt{true} = (w_4 \mid\!- w_1 = w_2 \texttt{ if } w_3),$$

then

$$T \vdash t_1 = t_2.$$

An analogous result holds when $(w_4 \mid\!- w_1 = w_2 \texttt{ if } w_3)$ is replaced with $(w_4 \mid\!- w_1 : \overline{s} \texttt{ if } w_3)$.

Proof. By structural induction on the proof in MEL. According to the last rule of deduction used, we have:

– **(Reflexivity)** and **(Membership)**. They are not possible.
– **(Symmetry)**. By the induction hypothesis.
– **(Transitivity)**. Suppose we have $\mathrm{U}_{\mathrm{MEL}} \vdash (w_4 \mid\!- w_1 = w_2 \texttt{ if } w_3) = \texttt{true}$ (the symmetric case is analogous). There is a term m over the signature of $\mathrm{U}_{\mathrm{MEL}}$ such that

$$\frac{\mathrm{U}_{\mathrm{MEL}} \vdash (w_4 \mid\!- w_1 = w_2 \texttt{ if } w_3) = m \qquad \mathrm{U}_{\mathrm{MEL}} \vdash m = \texttt{true}}{\mathrm{U}_{\mathrm{MEL}} \vdash (w_4 \mid\!- w_1 = w_2 \texttt{ if } w_3) = \texttt{true}}.$$

  By Lemma 1 applied to the proof δ of $(w_4 \mid\!- w_1 = w_2 \texttt{ if } w_3) = m$ we have one of the following alternatives:

  1. m is $w_4' \mid\!- w_1' = w_2' \texttt{ if } w_3'$, with derivations for $\mathrm{U}_{\mathrm{MEL}} \vdash \overline{t_1} = w_1'$, $\mathrm{U}_{\mathrm{MEL}} \vdash \overline{t_2} = w_2'$, $\mathrm{U}_{\mathrm{MEL}} \vdash \texttt{none} = w_3'$, and $\mathrm{U}_{\mathrm{MEL}} \vdash \overline{T} = w_4'$: the result follows by applying induction hypothesis to the proof of $m = \texttt{true}$.
  2. There is a proof of $(w_4 \mid\!- w_1 = w_2 \texttt{ if } w_3) = \texttt{true}$ whose depth is less than or equal to that of δ and thus less than that of the original derivation, and we apply the induction hypothesis again.

– **(Replacement)**. Note that the context C used for this rule must have been the empty one, because there is none around `true`. Thus, the only equations that can have been applied are the ones corresponding to the rules of deduction of MEL in Fig. 1; let us consider each case separately:
  - `reflexivity`. In this case w_1 is syntactically equal to w_2, hence $\mathrm{U}_{\mathrm{MEL}} \vdash \overline{t_1} = \overline{t_2}$ and the result follows by Proposition 5.
  - `replacement`. By Proposition 7, there is an axiom $t = t'$ **if** *Cond* (resp. $t : s$ **if** *Cond*) in E such that all atomic formulae in $\sigma(\mathit{Cond})$ can be proved in T. In this case, w_1 is $\texttt{applyC}(\overline{C}, \texttt{applyS}(\overline{\sigma}, \overline{t}))$, w_2 is $\texttt{applyC}(\overline{C}, \texttt{applyS}(\overline{\sigma}, \overline{t'}))$, and w_3 is `none`, and, by Propositions 3 and 4, t_1 is $C[\sigma(t)]$ and t_2 is $C[\sigma(t')]$. But then $T \vdash t_1 = t_2$ follows by **(Replacement)**.
  - `symmetry`. By induction hypothesis and **(Symmetry)**.
  - `transitivity`. The result follows by Proposition 6, the induction hypothesis, and **(Transitivity)**.
  - `membership`. The result follows by Proposition 6, the induction hypothesis, and **(Membership)**. □

As a corollary of the reflective results proved in this section, we will show in the next two sections the reflective nature of two other related logics: many-sorted equational logic and Horn logic with equality.

4. Reflection in many-sorted equational logic

Many-sorted equational logic — in short, MSEL — is a sublogic of MEL, namely the sublogic obtained by making the set of sorts empty [43] (and renaming "kind" as "sort"); in particular, for all theories T in MSEL, and sentences ϕ over the signature of T, it holds that $T \vdash_{\mathrm{MSEL}} \phi \iff T \vdash_{\mathrm{MEL}} \phi$. Note that, since we have only used kinds and conditional equations not involving any memberships in the definition of $\mathrm{U}_{\mathrm{MEL}}$, we have that $\mathrm{U}_{\mathrm{MEL}}$ is a theory in MSEL, which turns out to be reflective.

Theorem 4. $\mathrm{U}_{\mathrm{MEL}}$ *is a universal theory in* MSEL *for the class of finitely presentable theories having nonempty sorts.*

Proof. For all finitely presentable theories T in MSEL having nonempty sorts and sentences ϕ over the signature of T,

$$T \vdash_{\mathrm{MSEL}} \phi \iff T \vdash_{\mathrm{MEL}} \phi \iff \mathrm{U}_{\mathrm{MEL}} \vdash_{\mathrm{MEL}} \overline{T \vdash \phi} \iff \mathrm{U}_{\mathrm{MEL}} \vdash_{\mathrm{MSEL}} \overline{T \vdash \phi}$$

since, by definition, $\overline{T \vdash \phi}$ is a sentence in MSEL. □

5. Reflection in Horn logic with equality

In [43] it is shown that MEL is equivalent to many-sorted Horn logic with equality—in short, $\mathrm{MSHORN}^{=}$. It is not surprising then that the reflective results about MEL can be translated straightforwardly to $\mathrm{MSHORN}^{=}$.

A signature in $\mathrm{MSHORN}^{=}$ is a triple (L, Σ, Π), with L a set of sorts, $\Sigma = \{\Sigma_{w,l}\}_{(w,l)\in L^*\times L}$ a family of sets of function symbols, and $\Pi = \{\Pi_w\}_{w\in L^*}$ a family of sets of predicate symbols. A signature $\Omega = (K, \Sigma, S)$ in MEL can then be mapped to a signature $\Omega^\star = (K, \Sigma, S^\star)$ in $\mathrm{MSHORN}^{=}$ by taking $S^\star_k$ to be the set S_k for $k \in K$, and $S^\star_w = \emptyset$ for any $w \in K^* \setminus K$. Thus, if we adopt a postfix notation $\_ : s$ for each predicate in $\Omega^\star$, corresponding to a sort s in Ω, each sentence over Ω in MEL can be seen as a sentence over $\Omega^\star$ in $\mathrm{MSHORN}^{=}$. We will write $(\Omega, E)^\star$ for $(\Omega^\star, E)$. Then, for all sentences ϕ over Ω it holds that $(\Omega, E) \vdash_{\mathrm{MEL}} \phi \iff (\Omega, E)^\star \vdash_{\mathrm{MSHORN}^{=}} \phi$.

Moreover, in [43] a translation from $\mathrm{MSHORN}^{=}$ to MEL is also defined. A signature (L, Σ, Π) in $\mathrm{MSHORN}^{=}$ is mapped to a theory $J(L, \Sigma, \Pi)$ in MEL whose signature consists of:

- a set of kinds $K = L \uplus \{p(w) \mid w \in L^* \setminus L \text{ and } \Pi_w \neq \emptyset\}$;
- for each kind $k \in K$, the set of sorts S_k is Π_k if $k \in L$ or Π_w if k is $p(w)$;
- a set of operators

$$\begin{aligned}\Delta = \Sigma \cup \{\langle \_, \ldots, \_\rangle : l_1 \ldots l_n \longrightarrow p(l_1, \ldots, l_n) \mid p(l_1, \ldots, l_n) \in K \setminus L\} \\ \cup \{\pi_i : p(l_1, \ldots, l_n) \longrightarrow l_i \mid 1 \leq i \leq n,\, p(l_1, \ldots, l_n) \in K \setminus L\}.\end{aligned}$$

The idea is to represent the cartesian product of the kinds $l_1, \ldots, l_n$ by means of the kind $p(l_1, \ldots, l_n)$. Thus, the axioms of the theory $J(L, \Sigma, \Pi)$ are

$$\begin{aligned}&(\forall x_1 : l_1, \ldots, x_n : l_n)\ \pi_i(\langle x_1, \ldots, x_n\rangle) = x_i \qquad (1 \leq i \leq n)\\ &(\forall y : p(l_1, \ldots, l_n))\ y = \langle \pi_1(y), \ldots, \pi_n(y)\rangle\end{aligned}$$

for every $p(l_1, \ldots, l_n)$.

The translation α of sentences leaves each equation $t = t'$ unchanged and maps each atomic formula $P(t_1, \ldots, t_n)$ to the membership assertion $\langle t_1, \ldots, t_n\rangle : P$, and each Horn clause

$$(\forall X)\ at \Leftarrow u_1 = v_1 \wedge \cdots \wedge u_n = v_n \wedge P_1(\overline{w_1}) \wedge \cdots \wedge P_m(\overline{w_m})$$

to the sentence

$$(\forall X)\ \alpha(at)\ \textbf{if}\ u_1 = v_1 \wedge \cdots \wedge u_n = v_n \wedge \langle \overline{w_1}\rangle : P_1 \wedge \cdots \wedge \langle \overline{w_m}\rangle : P_m.$$

A theory $T = (L, \Sigma, \Pi, \Gamma)$ in $\mathrm{MSHORN}^{=}$ is then mapped to the theory $J(T)$ that results from adding to $J(L, \Sigma, \Pi)$ the translation of the clauses in Γ. In [43] it is proven that $T \vdash_{\mathrm{MSHORN}^{=}} \phi \iff J(T) \vdash_{\mathrm{MEL}} \alpha(\phi)$ and we have the following result.

Theorem 5. $\mathrm{U}^\star_{\mathrm{MEL}}$ *is a universal theory in* $\mathrm{MSHORN}^{=}$ *for the class of all finitely presentable theories with nonempty sorts.*

Proof. For all finitely presentable theories with nonempty sorts T in $\mathrm{MSHORN}^{=}$, and sentences ϕ over T,

$$\begin{aligned}T \vdash_{\mathrm{MSHORN}^{=}} \phi &\iff J(T) \vdash_{\mathrm{MEL}} \alpha(\phi)\\ &\iff \mathrm{U}_{\mathrm{MEL}} \vdash_{\mathrm{MEL}} \overline{J(T) \vdash_{\mathrm{MEL}} \alpha(\phi)}\\ &\iff \mathrm{U}^\star_{\mathrm{MEL}} \vdash_{\mathrm{MSHORN}^{=}} \overline{J(T) \vdash_{\mathrm{MEL}} \alpha(\phi)}.\end{aligned}$$

□

6. Reflection in rewriting logic

Rewriting logic — in short, RL — is parameterized with respect to an underlying equational logic; here we use MEL, which is the most general equational logic studied so far as a parameter of rewriting logic. Given a MEL signature $\Omega = (K, \Sigma, S)$, the *sentences* of rewriting logic are "sequents" of the form $t \longrightarrow t'$, where t and t' are Ω-terms of the same kind possibly involving some variables from a K-kinded set of variables X.

A *rewrite theory* T is a 3-tuple $T = (\Omega, E, R)$, where (Ω, E) is a MEL theory and R is a set of *(conditional) rewrite rules* of the form

$$t \longrightarrow t' \textbf{ if } \bigwedge_{i\in I}(u_i : s_i) \wedge \bigwedge_{j\in J}(v_j = v'_j) \wedge \bigwedge_{l\in L}(w_l \longrightarrow w'_l),$$

with $p_i = q_i$ and $w_j : s_j$ atomic formulae in membership equational logic for $i \in I$ and $j \in J$, and for appropriate kinds k and k_l, $t, t' \in T_{\Sigma,k}(\vec{x})$, and $w_l, w'_l \in T_{\Sigma,k_l}(\vec{x})$ for $l \in L$. As in Section 3, we assume that the underlying MEL theory has nonempty kinds.

The rules of RL**.** We now introduce the rules of deduction of RL. Our formulation follows that in [7], that generalizes [41] by giving an explicit rule of E-equality for rewrite sequents $t \longrightarrow t'$ instead of absorbing such a rule in sequents $[t] \longrightarrow [t']$ between E-equivalence classes. But as we did for membership equational logic, the congruence rule is removed and subsumed in the **(Replacement)** rule.

Given a rewrite theory $T = (\Omega, E, R)$, we say that T *entails* a sequent $t \longrightarrow t'$ and write $T \vdash t \longrightarrow t'$ if and only if $t \longrightarrow t'$ can be obtained by finite application of the following *rules of deduction*:

1. **Reflexivity**. For every $t \in T_\Sigma(X)$,
$$\frac{}{t \longrightarrow t}.$$
2. **Replacement**. For each rewrite rule $(t \longrightarrow t' \text{ if } C_{mb} \wedge C_{eq} \wedge C_{rl})$ in R, with t, t' of kind k, context $C^k \in \mathrm{C}^{\iota}_{\Sigma}(X)$, and substitution σ, where $C_{mb} \triangleq (u_1 : s_1 \wedge \cdots \wedge u_j : s_j)$, $C_{eq} \triangleq (v_1 = v'_1 \wedge \cdots \wedge v_k = v'_k)$, and $C_{rl} \triangleq (w_1 \longrightarrow w'_1 \wedge \cdots \wedge w_h \longrightarrow w'_h)$,
$$\frac{\begin{array}{cccc}(\Omega, E) \vdash \sigma(u_1) : s_1 & \cdots & (\Omega, E) \vdash \sigma(u_j) : s_j \\ (\Omega, E) \vdash \sigma(v_1) = \sigma(v'_1) & \cdots & (\Omega, E) \vdash \sigma(v_k) = \sigma(v'_k) \\ \sigma(w_1) \longrightarrow \sigma(w'_1) & \cdots & \sigma(w_h) \longrightarrow \sigma(w'_h)\end{array}}{C[\sigma(t)] \longrightarrow C[\sigma(t')]}.$$
3. **Transitivity**. For every $t, t', t'' \in T_\Sigma(X)$,
$$\frac{t \longrightarrow t' \qquad t' \longrightarrow t''}{t \longrightarrow t''}.$$
4. **Equality**. For every $t, t', u, u' \in T_\Sigma(X)$,
$$\frac{(\Omega, E) \vdash t = u \qquad (\Omega, E) \vdash t' = u' \qquad t \longrightarrow t'}{u \longrightarrow u'}.$$

6.1. A universal theory for RL

Here, we introduce the universal theory U_{RL} and a representation function $\overline{\_ \vdash \_}$ that encodes pairs consisting of a theory T and a sentence over its signature as a sentence in U_{RL}, and prove the universality of U_{RL}. The key observation is that U_{RL} is an extension of $\mathrm{U}_{\mathrm{MEL}}$, so that we can use the universality of $\mathrm{U}_{\mathrm{MEL}}$ in the proof of the universality of U_{RL}. In what follows, we will be dealing with *finitely presentable* theories in RL.

The signature of U_{RL}**.** The signature of the theory U_{RL} is an extension of the signature of $\mathrm{U}_{\mathrm{MEL}}$. To represent (possibly conditional) rules, the signature of U_{RL} includes the operators:

```
op _=>_ : [Term] [Term] -> [AtomR] .
op noneR : -> [RuleCondition] .
op _/\_ : [Atom] [RuleCondition] -> [RuleCondition] .
op _/\_ : [AtomR] [RuleCondition] -> [RuleCondition] .
op _=>_if_ : [Term] [Term] [RuleCondition] -> [Rule] .
```

Theories are represented using:

```
op (_,_,_) : [Signature] [AxiomSet] [RuleSet] -> [RLTheory] .
```

Finally, the signature of U_{RL} contains a Boolean operator

```
op _|-_ : [RLTheory] [AtomR] -> [Bool] .
```

to represent provability of sentences in a given rewrite theory; the main axioms of U_{RL} define this operator.

As happened with $\mathrm{U}_{\mathrm{MEL}}$, there are also a number of auxiliary operators. There are operators for sets of rewrite rules

```
op emptyR : -> [RuleSet] .
op unionR : [Rule] [RuleSet] -> [RuleSet] .
```

as well as operations to check satisfaction of a condition in a given theory:

```
op satisfyRC : [RLTheory] [RuleCondition] [Substitution] -> [Bool] .
op satisfyR : [RLTheory] [Rule] [Substitution] -> [Bool] .
```

The representation function. We next define the representation function $\overline{\_ \vdash \_}$. For all finitely presentable rewrite theories T with nonempty kinds, and sentences $t \longrightarrow t'$ over the signature of T,

$$\overline{T \vdash t \longrightarrow t'} \triangleq (\overline{T} \,\texttt{|-}\, \overline{t} \,\texttt{=>}\, \overline{t'}) \longrightarrow \texttt{true}.$$

where $\overline{(\_)}$ is an extension of the representation function for $\mathrm{U}_{\mathrm{MEL}}$, defined in the expected way.

1. For a theory $T = (\Omega, E, R)$,
$$\overline{T} \triangleq (\overline{\Omega}, \overline{E}, \overline{R}).$$
2. For R a set of rewrite rules $\{r_1, \ldots, r_n\}$,
$$\overline{R} \triangleq \texttt{consR}(\overline{r_1}, \ldots, \texttt{consR}(\overline{r_n}, \texttt{emptyR}) \ldots).$$
3. For a rewrite $t \longrightarrow t'$,
$$\overline{t \longrightarrow t'} \triangleq \overline{t} \,\texttt{=>}\, \overline{t'}.$$
4. For *Cond* a conjunction of equations, memberships, and rewrites $A_1 \wedge A_2 \wedge \cdots \wedge A_n$,
$$\overline{Cond} \triangleq \overline{A_1} \,\texttt{/\textbackslash}\, (\overline{A_2} \,\texttt{/\textbackslash}\, \cdots \,\texttt{/\textbackslash}\, (\overline{A_n} \,\texttt{/\textbackslash}\, \texttt{noneR}) \ldots).$$
5. For r a rewrite rule $t \longrightarrow t'$ **if** $A_1 \wedge \cdots \wedge A_n$,
$$\overline{r} \triangleq \overline{t} \,\texttt{=>}\, \overline{t'} \,\texttt{if}\, \overline{A_1 \wedge \cdots \wedge A_n}.$$

The axioms of U_{RL}. Finally we define the axioms of U_{RL}, which include rules that correspond to the inference rules of RL, and all the equations in $\mathrm{U}_{\mathrm{MEL}}$. The same remarks as in 'The axioms of $\mathrm{U}_{\mathrm{MEL}}$' apply to our notation for these rules.

In what follows, we assume a finitely presentable rewrite theory with nonempty kinds, $T = (\Omega, E, R)$, where $\Omega = (K, \Sigma, S)$.

Again, we need equations between the set operators.

```
eq unionR(r̄, unionR(r̄',ĀS)) = unionR(r̄', unionR(r̄,ĀS)) .
eq unionR(r̄, unionR(r̄,ĀS)) = unionR(r̄,ĀS) .
```

And analogously for the operator `_/\_` that builds conditions.

The equations for `satisfyRC`, like those for `satisfyC`, simply unfold the components of a condition.

```
eq satisfyRC(T̄,none,σ̄) = true .
eq satisfyRC(T̄,Ā /\ C̄,σ̄) = satisfyR(T̄,Ā,σ̄) and satisfyRC(T̄,C̄,σ̄) .

eq satisfyR(T̄,t̄ => t̄',σ̄) = (T̄ |- applyS(σ̄,t̄) => applyS(σ̄,t̄')) .
eq satisfyR(T̄,t̄ = t̄',σ̄) = (T̄ |- applyS(σ̄,t̄) = applyS(σ̄,t̄') if none) .
eq satisfyR(T̄,t̄ : s̄,σ̄) = (T̄ |- applyS(σ̄,t̄) : s̄ if none) .
```

```
*** reflexivity
rl ((Ω,E,R) |- t => t) => true.

*** replacement
crl ((Ω,E,R) |- applyC(C, applyS(σ,t)) => applyC(C, applyS(σ,t'))) => true
    if R = unionR(t = t' if Cond,R)
    /\ satisfyR((Ω,E,R),Cond,σ) = true .

*** transitivity
crl ((Ω,E,R) |- t => t') => true
    if parse(t'',Ω) = true
    /\ ((Ω,E,R) |- t => t'') => true
    /\ ((Ω,E,R) |- t'' => t') => true .

*** equality
crl ((Ω,E,R) |- u => u') => true
    if parse(t,Ω) = true
    /\ parse(t',Ω) = true
    /\ ((Ω,E) |- t = u if none) = true
    /\ ((Ω,E) |- t' = u' if none) = true
    /\ ((Ω,E,R) |- t => t') => true .
```

Fig. 2. The universal theory U_{RL} (fragment).

Proposition 8. *For all atomic formulae, rewrites $A_1, \ldots, A_n$, and substitutions σ, the following are equivalent:*

1. $\mathrm{U}_{\mathrm{RL}} \vdash \overline{T \vdash \sigma(A_i)}$ *for each i; that is, for each A_i of the form $t_i \longrightarrow t_i'$,*

$$\mathrm{U}_{\mathrm{RL}} \vdash (\overline{T} \,\texttt{|-}\, \overline{\sigma(t_i)} \,\texttt{=>}\, \overline{\sigma(t_i')}) \longrightarrow \texttt{true},$$

for each A_i of the form $t_i = t_i'$,

$$\mathrm{U}_{\mathrm{RL}} \vdash (\overline{T} \,\texttt{|-}\, \overline{\sigma(t_i)} = \overline{\sigma(t_i')} \,\texttt{if none}) = \texttt{true},$$

and for each A_i of the form $t_i : s_i$,

$$\mathrm{U}_{\mathrm{RL}} \vdash (\overline{T} \,\texttt{|-}\, \overline{\sigma(t_i)} : \overline{s_i} \,\texttt{if none}) = \texttt{true}.$$

2. *It holds that*

$$\mathrm{U}_{\mathrm{RL}} \vdash \texttt{satisfyRC}(\overline{T}, \overline{A_1 \wedge \cdots \wedge A_n}, \overline{\sigma}) = \texttt{true}.$$

Finally, the rules of deduction of RL are specified as shown in Fig. 2.

6.2. The correctness of the universal theory RL

We already have the necessary ingredients to show the correctness of the universal theory U_{RL}. The proof is analogous to that of Theorem 1, using in key steps of the proof the fact that U_{RL} is an extension of $\mathrm{U}_{\mathrm{MEL}}$, so we will not spell out all the details but focus on the most complex, interesting cases.

As happened in the previous section, some auxiliary lemmas are needed before the main result can be proven. In the following proofs we write $\mathrm{U}_{\mathrm{RL}} \vdash w = w'$ to denote that the equation $w = w'$ can be derived in the equational subtheory of U_{RL}.

Lemma 2. *For all terms w of kind* `[Bool]`*, w_1, w_2 of kind* `[Term]`*, and w_3 of kind* `[RLTheory]` *in* U_{RL} *such that*

$$\mathrm{U}_{\mathrm{RL}} \vdash w = (w_3 \,\texttt{|-}\, w_1 \,\texttt{=>}\, w_2) \quad \textit{or} \quad \mathrm{U}_{\mathrm{RL}} \vdash (w_3 \,\texttt{|-}\, w_1 \,\texttt{=>}\, w_2) = w,$$

the term w must of the form $w_3' \,\texttt{|-}\, w_1' \,\texttt{=>}\, w_2'$ for terms w_1', w_2', and w_3' such that $\mathrm{U}_{\mathrm{RL}} \vdash w_i = w_i'$, $1 \leq i \leq 3$.

Proof. By structural induction on the derivation. In the case of the **(Replacement)** rule note that no equations apply to the operator `_|-_=>_` representing derivability; the other cases are immediate. □

The following lemma is analogous to Lemma 1; now, *depth*(δ) refers to the depth of a derivation δ in rewriting logic.

Lemma 3. *For all terms t_1, t_2 over a rewrite theory T, and terms w_1, w_2 of kind* `[Term]`*, and w_3 of kind* `[RLTheory]` *in* U_{RL} *such that* $\mathrm{U}_{\mathrm{RL}} \vdash \overline{t_1} = w_1$, $\mathrm{U}_{\mathrm{RL}} \vdash \overline{t_2} = w_2$, *and* $\mathrm{U}_{\mathrm{RL}} \vdash \overline{T} = w_3$, *whenever there is a proof δ in* U_{RL} *of the rewrite*

$$(w_3 \,|\!-\, w_1 \,\texttt{=>}\, w_2) \longrightarrow m$$

for some term m of kind `[Bool]` *over the signature of* U_{RL}*, one of the following alternatives holds:*

1. *m is $w_3' \,|\!-\, w_1' \,\texttt{=>}\, w_2'$, for terms w_1', w_2', and w_3' such that* $\mathrm{U}_{\mathrm{RL}} \vdash \overline{t_1} = w_1'$, $\mathrm{U}_{\mathrm{RL}} \vdash \overline{t_2} = w_2'$, *and* $\mathrm{U}_{\mathrm{RL}} \vdash \overline{T} = w_3'$*; or*
2. *there exists a proof δ' of the rewrite $(w_3 \,|\!-\, w_1 \,\texttt{=>}\, w_2) \longrightarrow \texttt{true}$ such that $\mathit{depth}(\delta') \leq \mathit{depth}(\delta)$.*

Proof. As in Lemma 1, we proceed by structural induction on the proof δ. The only case which is treated in a different way is the one that corresponds to the **(Equality)** rule; hence, let us assume that the last step in the proof is

$$\frac{\mathrm{U}_{\mathrm{RL}} \vdash t = (w_3 \,|\!-\, w_1 \,\texttt{=>}\, w_2) \qquad \mathrm{U}_{\mathrm{RL}} \vdash t' = u' \qquad t \longrightarrow t'}{(w_3 \,|\!-\, w_1 \,\texttt{=>}\, w_2) \longrightarrow u'}.$$

By Lemma 2, t is of the form $w_3' \,|\!-\, w_1' \,\texttt{=>}\, w_2'$, with $\mathrm{U}_{\mathrm{RL}} \vdash w_i = w_i'$, $1 \leq i \leq 3$. By the induction hypothesis applied to the proof of $(w_3' \,|\!-\, w_1' \,\texttt{=>}\, w_2') \longrightarrow t'$, one of the following alternatives holds:

- t' is of the form $w_3'' \,|\!-\, w_1'' \,\texttt{=>}\, w_2''$, with $\mathrm{U}_{\mathrm{RL}} \vdash \overline{t_1} = w_1''$, $\mathrm{U}_{\mathrm{RL}} \vdash \overline{t_2} = w_2''$, and $\mathrm{U}_{\mathrm{RL}} \vdash \overline{T} = w_3''$. But, again by Lemma 2, then u' has the same form and (1) holds.
- There is a proof of $(w_3' \,|\!-\, w_1' \,\texttt{=>}\, w_2') \longrightarrow \texttt{true}$ whose depth is less than, or equal to, that of the proof of $(w_3' \,|\!-\, w_1' \,\texttt{=>}\, w_2') \longrightarrow t'$. But then, if we modify the original derivation of $(w_3 \,|\!-\, w_1 \,\texttt{=>}\, w_2) \longrightarrow u'$ by replacing the proof of $t \longrightarrow t'$ by that of $(w_3' \,|\!-\, w_1' \,\texttt{=>}\, w_2') \longrightarrow \texttt{true}$, and the proof of $\mathrm{U}_{\mathrm{RL}} \vdash t' = u'$ by $\mathrm{U}_{\mathrm{RL}} \vdash \texttt{true} = \texttt{true}$, we obtain a proof that satisfies (2). □

The main theorem is now proved by mimicking the proof of Theorem 1.

Theorem 6. *For all finitely presentable rewrite theories with nonempty kinds $T = (\Omega, E, R)$, with $\Omega = (K, \Sigma, S)$, and terms t, t' in $T_\Sigma(X)$*

$$T \vdash t \longrightarrow t' \iff \mathrm{U}_{\mathrm{RL}} \vdash (\overline{T} \,|\!-\, \overline{t} \,\texttt{=>}\, \overline{t'}) \longrightarrow \texttt{true}.$$

Proof. The proof in the ($\Rightarrow$)-direction follows the steps of that for Theorem 2: it proceeds by structural induction on the derivation of $T \vdash t \longrightarrow t'$ and the only new case is the one that corresponds to the **(Equality)** rule. Thus, let us assume that the last step in the derivation is of the form

$$\frac{(\Omega, E) \vdash t = u \qquad (\Omega, E) \vdash t' = u' \qquad t \longrightarrow t'}{u \longrightarrow u'}.$$

Since $\mathrm{U}_{\mathrm{MEL}}$ is universal we have $\mathrm{U}_{\mathrm{MEL}} \vdash ((\overline{\Omega}, \overline{E}) \,|\!-\, \overline{t} = \overline{u} \ \texttt{if none}) = \texttt{true}$ and also $\mathrm{U}_{\mathrm{MEL}} \vdash ((\overline{\Omega}, \overline{E}) \,|\!-\, \overline{t'} = \overline{u'} \ \texttt{if none}) = \texttt{true}$ and by the induction hypothesis, $\mathrm{U}_{\mathrm{RL}} \vdash (\overline{T} \,|\!-\, \overline{t} \,\texttt{=>}\, \overline{t'}) \longrightarrow \texttt{true}$; by Proposition 6, since U_{RL} contains $\mathrm{U}_{\mathrm{MEL}}$ we can apply **(Replacement)** with the `equality` rule to obtain $\mathrm{U}_{\mathrm{RL}} \vdash (\overline{T} \,|\!-\, \overline{u} \,\texttt{=>}\, \overline{u'}) \longrightarrow \texttt{true}$.

The ($\Leftarrow$)-direction follows from a generalization analogous to that in Theorem 3: for all terms w_1, w_2, and w_3 over the signature of U_{RL} such that they are provably equal to, respectively, $\overline{t_1}$, $\overline{t_2}$, and $\overline{T}$, if

$$\mathrm{U}_{\mathrm{RL}} \vdash (w_3 \,|\!-\, w_1 \,\texttt{=>}\, w_2) \longrightarrow \texttt{true},$$

then

$$T \vdash t_1 \longrightarrow t_2.$$

Again, the proof follows the same steps as in Theorem 3. The case for the **(Transitivity)** rule now relies on Lemma 3 and the one for **(Replacement)** introduces a new case, corresponding to the `equality` rule. In this last situation we

get the result from Proposition 6, the induction hypothesis, and Theorem 3 itself. In addition, we also have to consider the case in which the last rule applied has been **(Equality)**:

$$\frac{\mathrm{U}_{\mathrm{RL}} \vdash t = (w_3 \,|\!-\, w_1 \Rightarrow w_2) \qquad \mathrm{U}_{\mathrm{RL}} \vdash t' = \mathtt{true} \qquad t \longrightarrow t'}{(w_3 \,|\!-\, w_1 \Rightarrow w_2) \longrightarrow \mathtt{true}}.$$

By Lemma 2, t has the form $w_3' \,|\!-\, w_1' \Rightarrow w_2'$ with $\mathrm{U}_{\mathrm{RL}} \vdash w_i = w_i'$, $1 \leq i \leq 3$, and Lemma 3 can be applied to the proof of $t \longrightarrow t'$ to distinguish the following cases:

- t' has the form $w_3'' \,|\!-\, w_1'' \Rightarrow w_2''$. This situation cannot happen due to Lemma 2, because we have $\mathrm{U}_{\mathrm{RL}} \vdash t' = \mathtt{true}$.
- There is a proof of $(w_3' \,|\!-\, w_1' \Rightarrow w_2') \longrightarrow \mathtt{true}$ whose depth is less than, or equal to, that of $t \longrightarrow t'$ and therefore less than the original proof; the result is again obtained by the induction hypothesis. □

7. Comparison with previous results

The work discussed in this paper generalizes and extends our previous work on reflection in rewriting logic [10,17,45]. The results presented here *generalize* our previous results on reflection in rewriting logic to its more general variant, namely, to the case of conditional rewrite theories whose underlying equational specifications are theories in membership equational logic. To simplify the correctness proof of the universal theory, we have, however, adopted a different approach for defining the universal theory. Essentially, in [10,17], an entailment of the form $T \vdash t \longrightarrow t'$ was reflected as $U \vdash \langle \overline{T}, \overline{t} \rangle \longrightarrow \langle \overline{T}, \overline{t}' \rangle$. Accordingly, the "transitivity" rule of deduction did not have to be explicitly reified in the universal theory since, if $T \vdash t_1 \longrightarrow t_3$ was proved by transitivity from $T \vdash t_1 \longrightarrow t_2$ and $T \vdash t_2 \longrightarrow t_3$, then, of course, $U \vdash \langle \overline{T}, \overline{t}_1 \rangle \longrightarrow \langle \overline{T}, \overline{t}_3 \rangle$, would also be proved by transitivity from $U \vdash \langle \overline{T}, \overline{t}_1 \rangle \longrightarrow \langle \overline{T}, \overline{t}_2 \rangle$ and $U \vdash \langle \overline{T}, \overline{t}_2 \rangle \longrightarrow \langle \overline{T}, \overline{t}_3 \rangle$. In our current approach, however, an entailment of the form $T \vdash t \longrightarrow t'$ is reflected as $U \vdash (\overline{T} \,|\!-\, \overline{t} \Rightarrow \overline{t}') \longrightarrow \mathtt{true}$, and the "transitivity" rule (and also the "symmetry" rule in the case of membership equational logic) has to be explicitly reflected in the universal theory. The original approach corresponds to thinking of the universal theory from a computational point of view, with the transitivity rule of deduction in charge of applying successive reductions; this complicated the proofs, since we had no control over that process. The new approach, however, is more logical and results in a much simpler, easy to follow scheme, with shorter and less tedious proofs.

In addition, the results presented here *extend* in a natural way our previous results on reflection in rewriting logic to other related logics, namely, membership equational logic, many-sorted equational logic, and Horn logic with equality. The extensions are very natural, in the sense that the proposed universal theories are themselves related.

8. Conclusion

We have shown that the generalized variant of rewriting logic where the underlying equational specifications are membership equational theories, and where the rules are conditional and can have equations, memberships and rewrites in the conditions, is reflective. We have also shown that membership equational logic, many-sorted equational logic, and Horn logic with equality are likewise reflective. These results provide logical foundations for reflective languages and tools based on these logics, and in particular for the Maude language itself. The results presented here can be further developed and generalized in several directions, including:

- giving proofs of reflection for other more restrictive but frequently used logics, such as Horn logic without equality;
- further extending the rewriting logic results to theories where some of the operators are *frozen* [7], so that no rewriting is allowed under them, and to theories where some kinds can be empty;
- developing adequate strategies to execute the universal theories of rewriting logic and of membership equational logic in Maude, so that proof objects can be associated to reflective proofs when desired.

This work is one step further within a broader effort, whose first results appeared in [16], to develop a general theory of reflection for logics and declarative languages. In this regard, the results presented in this paper raise the issue of how the universal theories of related logics are themselves related. To address in a precise and formalism-independent way this question, we expect that the metalogical foundations provided by the theory of general logics [40], which are at the base of our axiomatic approach to the study of reflection, can be very helpful. A different approach that we also intend to explore, and that was recently suggested to us by Prof. Mario Rodríguez-Artalejo, consists in the use of

Smullyan's elementary formal systems (EFS) [48]. The idea is to build an entailment system of all EFSs and to use it as an "intermediary" between the entailment systems of related logics, taking advantage of the fact that all recursively enumerable sets can be recognized by EFSs and that the set of derivable sentences in many sensible logics is in fact recursively enumerable.

Acknowledgments

We thank Narciso Martí-Oliet for many discussions on the topic of reflection in rewriting logic, and for his detailed and very helpful comments on earlier drafts of this paper. We also thank Mario Rodríguez-Artalejo for his insightful comments and suggestions on the issue of metalogical characterization of reflection.

References

[1] K. Asai, Reflecting on the metalevel interpreter written in direct style, in: International Lisp Conference 2003, ILC 2003, http://pllab.is.ocha.ac.jp/~asai/papers/papers.html.
[2] J.V. Baalen, J.L. Caldwell, S. Mishra, Specifying and checking fault-tolerant agent-based protocols using Maude, in: First Goddard Workshop on Formal Approaches to Agent-Based Systems, 5–7 April 2000, Greenbelt, MD, USA, Proceedings, in: Lecture Notes in Computer Science, vol. 1871, Springer-Verlag, 2000, pp. 180–193.
[3] D. Basin, M. Clavel, J. Meseguer, Reflective metalogical frameworks, ACM Transactions on Computational Logic 5 (3) (2004) 528–576.
[4] P. Borovanský, C. Kirchner, H. Kirchner, P.-E. Moreau, ELAN from a rewriting logic point of view, Theoretical Computer Science 285 (2) (2002) 155–185.
[5] A. Bouhoula, J.-P. Jouannaud, J. Meseguer, Specification and proof in membership equational logic, Theoretical Computer Science 236 (2000) 35–132.
[6] R.S. Boyer, J.S. Moore, Metafunctions: Proving them correct and using them efficiently as new proof procedures, in: R. Boyer, J. Moore (Eds.), The Correctness Problem in Computer Science, Academic Press, 1981, pp. 103–185.
[7] R. Bruni, J. Meseguer, Generalized rewrite theories, in: J.C.M. Baeten, J.K. Lenstra, J. Parrow, G.J. Woeginger (Eds.), Automata, Languages and Programming. 30th International Colloquium, ICALP 2003, June 30–July 4 2003, Eindhoven, The Netherlands, Proceedings, in: Lecture Notes in Computer Science, vol. 2719, Springer-Verlag, 2003, pp. 252–266.
[8] W. Cazzola, S. Chiba, G. Saake, RAM-SE'04. ECOOP'2004 Workshop on Reflection, AOP and Meta-Data for Software Evolution, http://www.disi.unige.it/person/CazzolaW/RAM-SE04.html.
[9] G. Ciobanu, D. Lucanu, Communicating concurrent objects in hidden CCS, in: N. Martí-Oliet (Ed.), Proceedings Fifth International Workshop on Rewriting Logic and its Applications, WRLA'04, 27–28 March 2004, Barcelona, Spain, in: Electronic Notes in Theoretical Computer Science, Elsevier, 2004, pp. 329–347.
[10] M. Clavel, Reflection in Rewriting Logic: Metalogical Foundations and Metaprogramming Applications, CSLI Publications, 2000.
[11] M. Clavel, F. Durán, S. Eker, P. Lincoln, N. Martí-Oliet, J. Meseguer, J.F. Quesada, Maude: Specification and programming in rewriting logic, Theoretical Computer Science 285 (2) (2002) 187–243.
[12] M. Clavel, F. Durán, S. Eker, P. Lincoln, N. Martí-Oliet, J. Meseguer, C. Talcott, Maude manual (version 2.2), http://maude.cs.uiuc.edu/manual/, 2005.
[13] M. Clavel, F. Durán, S. Eker, P. Lincoln, N. Martí-Oliet, J. Meseguer, C.L. Talcott, All About Maude, A High-Performance Logical Framework, in: Lecture Notes in Computer Science, vol. 4350, Springer-Verlag, 2007.
[14] M. Clavel, F. Durán, S. Eker, J. Meseguer, M.-O. Stehr, Maude as a formal meta-tool, in: J.M. Wing, J. Woodcock, J. Davies (Eds.), FM'99 — Formal Methods, World Congress on Formal Methods in the Development of Computing Systems, 20–24 September 1999, Toulouse, France, Proceedings, Volume II, in: Lecture Notes in Computer Science, vol. 1709, Springer-Verlag, 1999, pp. 1684–1703.
[15] M. Clavel, N. Martí-Oliet, M. Palomino, Formalizing and proving semantic relations between specifications by reflection, in: C. Rattray, S. Maharaj, C. Shankland (Eds.), Algebraic Methodology and Software Technology. 10th International Conference, AMAST 2004, 12–16 July 2004, Stirling, Scotland, UK, Proceedings, in: Lecture Notes in Computer Science, vol. 3116, Springer-Verlag, 2004, pp. 72–86.
[16] M. Clavel, J. Meseguer, Axiomatizing reflective logics and languages, in: G. Kiczales (Ed.), Proceedings of Reflection'96, April 1996, San Francisco, CA, 1996, pp. 263–288.
[17] M. Clavel, J. Meseguer, Reflection in conditional rewriting logic, Theoretical Computer Science 285 (2) (2002) 245–288.
[18] M. Clavel, M. Palomino, A. Riesco, Introducing the ITP tool: A tutorial, in: F. López-Fraguas (Ed.), PROLE'05. Selected papers, Journal of Universal Computer Science 12 (11) (2006) 1618–1650.
[19] P. Cointe (Ed.), Meta-Level Architectures and Reflection, Second International Conference, Reflection'99 Saint-Malo, 19–21 July 1999, France, Proceedings, in: Lecture Notes in Computer Science, vol. 1616, Springer-Verlag, 1999.
[20] G. Denker, C. Talcott (Eds.), Proceedings Sixth International Workshop on Rewriting Logic and its Applications, WRLA'06, 1–2 April 2006, Vienna, Austria, in: Electronic Notes in Theoretical Computer Science, Elsevier, 2006.
[21] R. Douence, M. Südholt, A generic reification technique for object-oriented reflective languages, Higher-Order and Symbolic Computation 14 (1) (2001) 7–34.
[22] F. Durán, A Reflective Module Algebra with Applications to the Maude Language, Ph.D. Thesis, Universidad de Málaga, Spain, June 1999, http://maude.cs.uiuc.edu/papers.

[23] F. Durán, The extensibility of Maude's module algebra, in: T. Rus (Ed.), Algebraic Methodology and Software Technology, 8th International Conference, AMAST 2000, 20–27 May 2000, IA City, Iowa, USA, Proceedings, in: Lecture Notes in Computer Science, vol. 1816, Springer-Verlag, 2000, pp. 422–437.
[24] F. Durán, S. Escobar, S. Lucas, New evaluation commands for Maude within Full Maude, in: N. Martí-Oliet (Ed.), Proceedings Fifth International Workshop on Rewriting Logic and its Applications, WRLA'04, 27–28 March 2004, Barcelona, Spain, in: Electronic Notes in Theoretical Computer Science, Elsevier, 2004, pp. 245–266.
[25] F. Durán, J. Meseguer, A Church-Rosser checker tool for Maude equational specifications, http://maude.cs.uiuc.edu/tools, 2000.
[26] B. Fischer, G. Roşu, Interpreting abstract interpretations in membership equational logic, in: M. van den Brand, R. Verma (Eds.), Proceedings of the International Workshop on Rule-Based Programming, RULE'01, in: Electronic Notes in Theoretical Computer Science, vol. 59, Elsevier, 2001.
[27] K. Futatsugi (Ed.), Proceedings Third International Workshop on Rewriting Logic and its Applications, WRLA'00, 10–20 September 2000, Kanazawa, Japan, in: Electronic Notes in Theoretical Computer Science, vol. 36, Elsevier, 2000, http://www.elsevier.com/locate/entcs/volume36.html.
[28] K. Futatsugi, R. Diaconescu, CafeOBJ Report, in: AMAST Series, World Scientific, 1998.
[29] F. Gadducci, U. Montanari (Eds.), Proceedings Fourth International Workshop on Rewriting Logic and its Applications, WRLA'02, 19–21 September 2002, Pisa, Italy, in: Electronic Notes in Theoretical Computer Science, vol. 71, Elsevier, 2002, http://www.elsevier.com/locate/entcs/volume71.html.
[30] F. Giunchiglia, P. Traverso, A. Cimatti, P. Pecchiari, A system for multi-level reasoning, in: Proceedings IMSA'92 — International Workshop on Reflection and Meta-Level Architecture, Information-Technology Promotion Agency, Japan, 1992, pp. 190–195.
[31] J. Goguen, J. Meseguer, Completeness of many-sorted equational logic, Houston Journal of Mathematics 11 (3) (1985) 307–334.
[32] P.M. Hill, J.W. Lloyd, The Gödel Programming Language, MIT Press, 1994.
[33] G. Kiczales, J. des Rivieres, D.G. Bobrow, The Art of the Metaobject Protocol, MIT Press, 1991.
[34] C. Kirchner, H. Kirchner (Eds.), Proceedings Second International Workshop on Rewriting Logic and its Applications, WRLA'98, 1–4 September 1998, Pont-à-Mousson, France, in: Electronic Notes in Theoretical Computer Science, vol. 15, Elsevier, 1998, http://www.elsevier.com/locate/entcs/volume15.html.
[35] N. Martí-Oliet (Ed.), Proceedings Fifth International Workshop on Rewriting Logic and its Applications, WRLA'04, 27–28 March 2004, Barcelona, Spain, in: Electronic Notes in Theoretical Computer Science, Elsevier, 2004.
[36] N. Martí-Oliet, J. Meseguer, Rewriting logic as a logical and semantic framework, in: D. Gabbay (Ed.), Handbook of Philosophical Logic, vol. 9, second edn, Kluwer, 2002, pp. 1–81.
[37] N. Martí-Oliet, J. Meseguer, Rewriting logic: Roadmap and bibliography, Theoretical Computer Science 285 (2) (2002) 121–154.
[38] N. Martí-Oliet, J. Meseguer, A. Verdejo, Towards a strategy language for Maude, in: N. Martí-Oliet (Ed.), Proceedings Fifth International Workshop on Rewriting Logic and its Applications, WRLA'04, 27–28 March 2004, Barcelona, Spain, in: Electronic Notes in Theoretical Computer Science, Elsevier, 2004, pp. 391–414.
[39] S. Matsuoka, T. Watanabe, Y. Ichisugi, A. Yonezawa, Object-oriented concurrent reflective architectures, in: M. Tokoro, O. Nierstrasz, P. Wegner (Eds.), Object-Based Concurrent Computing, in: Lecture Notes in Computer Science, vol. 612, Springer-Verlag, 1992, pp. 211–226.
[40] J. Meseguer, General logics, in: H.-D. Ebbinghaus, J. Fernández-Prida, M. Garrido, D. Lascar, M. Rodríguez-Artalejo (Eds.), Logic Colloquium'87, North-Holland, 1989, pp. 275–329.
[41] J. Meseguer, Conditional rewriting logic as a unified model of concurrency, Theoretical Computer Science 96 (1) (1992) 73–155.
[42] J. Meseguer (Ed.), Proceedings First International Workshop on Rewriting Logic and its Applications, WRLA'96, 3–6 September 1996, Asilomar, CA, in: Electronic Notes in Theoretical Computer Science, vol. 4, Elsevier, 1996, http://www.elsevier.com/locate/entcs/volume4.html.
[43] J. Meseguer, Membership algebra as a logical framework for equational specification, in: F. Parisi-Presicce (Ed.), Recent Trends in Algebraic Development Techniques, 12th International Workshop, WADT'97, 3–7 June 1997, Tarquinia, Italy, Selected Papers, in: Lecture Notes in Computer Science, vol. 1376, Springer-Verlag, 1998, pp. 18–61.
[44] P.C. Ölveczky, J. Meseguer, Specification of real-time and hybrid systems in rewriting logic, Theoretical Computer Science 285 (2) (2002) 359–405.
[45] M. Palomino, Comparing Meseguer's rewriting logic with the logic CRWL, in: M. Hanus (Ed.), International Workshop on Functional and (Constraint) Logic Programming, WFLP 2001, Selected Papers, in: Electronic Notes in Theoretical Computer Science, vol. 64, Elsevier, 2001.
[46] N. Shankar, Metamathematics, Machines, and Gödel's Proof, Cambridge University Press, 1994.
[47] J.R. Shoenfield, Degrees of Unsolvability, North-Holland, 1971.
[48] R.M. Smullyan, Theory of formal systems, in: Annals of Mathematics Studies, Princeton University Press, 1961.
[49] M.-O. Stehr, Rewriting logic and type theory — From applications to unification, Ph.D. Thesis, Computer Science Department, University of Hamburg, Germany, 2002.
[50] P. Thati, K. Sen, N. Martí-Oliet, An executable specification of asynchronous pi-calculus, in: F. Gadducci, U. Montanari (Eds.), Proceedings Fourth International Workshop on Rewriting Logic and its Applications, WRLA'02, 19–21 September 2002, Pisa, Italy, in: Electronic Notes in Theoretical Computer Science, vol. 71, Elsevier, 2002.
[51] V.F. Turchin, The concept of a supercompiler, ACM Transactions on Programming Languages and Systems 8 (3) (1986) 292–325.
[52] A. Verdejo, N. Martí-Oliet, Implementing CCS in Maude2, in: F. Gadducci, U. Montanari (Eds.), Proceedings Fourth International Workshop on Rewriting Logic and its Applications, WRLA'02, 19–21 September 2002, Pisa, Italy, in: Electronic Notes in Theoretical Computer Science, vol. 71, Elsevier, 2002.
[53] M. Wand, D.P. Friedman, The mystery of the tower revealed, Lisp and Symbolic Computation 1 (1) (1988) 11–38.
[54] A. Yonezawa, S. Matsuoka (Eds.), Metalevel Architectures and Separation of Crosscutting Concerns: Third International Conference, REFLECTION 2001, 25–28 September 2001, Kyoto, Japan, Proceedings, in: Lecture Notes in Computer Science, vol. 2192, Springer-Verlag, 2001.